UNIVERSIDADE FEDERAL DE OURO PRETO
ESCOLA DE MINAS
COLEGIADO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO - CECAU

DOUGLAS KENDI MIASHIRO

SISTEMA DE FISCALIZAÇÃO DE VAGAS PARA AUTOMÓVEIS

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Ouro Preto, 2018

DOUGLAS KENDI MIASHIRO

SISTEMA DE FISCALIZAÇÃO DE VAGAS PARA AUTOMÓVEIS

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. Alan Kardek Rêgo Segundo

Ouro Preto
Escola de Minas – UFOP
Fevereiro/2018

M618s Miashiro, Douglas Kendi.
Sistema de Fiscalização de Vagas para Automóveis [manuscrito] / Douglas Kendi Miashiro. - 2018.

68f.: il.: color; tabs.

Orientador: Prof. Dr. Alan Kardek Rêgo Segundo.

Monografia (Graduação). Universidade Federal de Ouro Preto. Escola de Minas. Departamento de Engenharia de Controle e Automação e Técnicas Fundamentais.

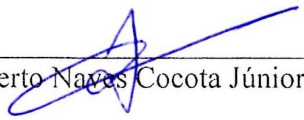
1. Deficientes - Serviços - Controle automático. 2. Sistemas Embarcados. 3. Microcontrolador. 4. Radiofrequência - RFID. I. Rêgo Segundo, Alan Kardek. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 681.5

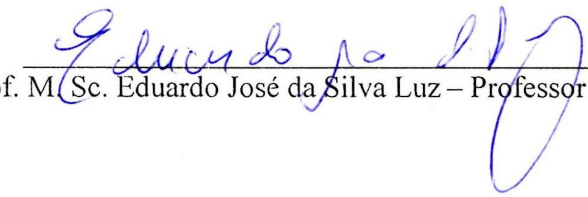
Catalogação: ficha@sisbin.ufop.br

Monografia defendida e aprovada, em 22 de fevereiro de 2018, pela comissão avaliadora constituída pelos professores:

Prof. Dr. Alan Kardek Rêgo Segundo - Orientador



Prof. Dr. José Alberto Naves Cocota Júnior – Professor Convidado



Prof. M. Sc. Eduardo José da Silva Luz – Professor Convidado

"A imaginação é mais importante que o conhecimento"
Albert Einstein

Resumo

A inclusão social é um conjunto de ações que tem o objetivo de assegurar os direitos das pessoas perante a sociedade, oferecendo formas de inclusão a aqueles que necessitam. No Brasil, já foram criadas algumas leis que garantem alguns direitos aos deficientes físicos e idosos, tais como as vagas especiais em estacionamentos públicos e privados, que são sinalizadas e de uso exclusivo desses membros. Elas são definidas de acordo com as melhores localizações no espaço físico, já que esses indivíduos podem encontrar dificuldades de locomoção. Um dos problemas enfrentados na utilização dessas vagas é o seu uso indevido por indivíduos que não se encaixam em nenhum desses grupos. O presente trabalho propõe a criação de um sistema automatizado, de baixo custo e de fácil implementação para a fiscalização das vagas especiais, utilizando a plataforma Arduino e uma comunicação wifi, com o objetivo de criar uma fiscalização visual e um sistema de assistência para as pessoas que utilizam essas vagas. O trabalho desenvolvido utiliza como referência alguns projetos e patentes já existentes, propondo alguma melhoria ou mudança em algum aspecto, com o intuito de adaptar o sistema para estacionamentos de espaços privados e de acordo com a realidade brasileira.

Palavras-chave: Fiscalização de vagas especiais, Sistemas Embarcados, Arduino, ESP8266, RFID.

Abstract

Social inclusion is a series of acts with the objective of ensuring the individual's rights, providings ways to include them in the society. In Brazil some laws were created to guarantee the rights of the people with disabilities as well as the elderly, such as exclusive parking spaces at public and private parking lots. These parking spaces are found on the best location in the physical space, since these people have locomotion dificulties. One of the recurrent problems at these places is the inproppriate use of these spots by people who do not fit in this group. The present project proposes the creation of a low-cost and simple implementation automate system to monitore these parking spaces using an Arduino platform and a Wi-Fi communication to create a visual inspection and, at the same time, a system which will assist those people who will use the exclusive parking spaces. The project uses as groundwork some existing patents, proposing a diferente variation, with the objective to adapt it to the Brazilian reality.

Keywords: Special Parking Spots Fiscalization, Embedded Systems, Arduino, RFID.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
AP	Access Point (Ponto de Acesso)
CECAU	Colegiado de Engenharia de Controle e Automação
CPU	Central Processing Unit (Unidade Central Processadora)
DC	Direct Current (corrente contínua)
DECAT	Departamento de Engenharia de Controle e Automação
DHCP	Dynamic Host Configuration Protocol (Protocolo de Configuração Dinâmica de Dost)
GND	Ground (Terra)
IP	Internet Protocol (Protocolo da Internet)
Led	Light Emitting Diode (Diodo Emissor de Luz)
RFID	Radio Frequency Identification (Identificador por Rádio Frequência)
STA	Station (Estação)
UFOP	Universidade Federal de Ouro Preto
USB	Universal Serial Bus
Vcc	Volts Corrente Contínua/ Alimentação de um equipamento

Sumário

1 Introdução

Até pouco tempo atrás, não haviam preocupações quanto a inclusão de idosos e deficientes físicos na sociedade. Com o passar do tempo, a sociedade foi se conscientizando quanto às problemáticas envolvendo essas pessoas, então ações foram sendo tomadas. A ONU criou um decreto tornando o ano de 1981 o Ano Internacional das Pessoas Portadoras de Deficiências (AIPPD) e, em 1999, foi declarado como Ano Internacional do Idoso.

No Brasil, foram criadas leis que asseguram os direitos e acessibilidade dessas pessoas, para seu maior conforto, já que elas precisam de mais comodidade e facilidade em muitos aspectos.

De acordo com o censo do IBGE (2010), mais de 13,2 milhões de pessoas tem algum grau de deficiência motora, equivalente a aproximadamente 7% dos brasileiros. Sendo assim, uma parcela considerável da população necessita de auxílio para ter acesso aos seus direitos, como qualquer outro cidadão.

Atualmente, para o uso das vagas da zona azul (vagas de uso exclusivo de idoso e deficientes físicos), o usuário deve ter uma autorização especial emitida pelo estado, a mesma deve estar visível no veículo quando estacionado em alguma dessas vagas.

Um dos grandes problemas enfrentados por deficientes físicos e idosos brasileiros, é a utilização dos direitos adquiridos por pessoas que não se encaixam em nenhum dos dois grupos, obstruindo o seu uso quando necessário, essa problemática se dá devido a falta de fiscalização dessas vagas (tanto em ambientes públicos para ambientes privados), fazendo com que os infratores saiam ilesos nessas situações. Este projeto apresenta a criação de um protótipo fiscalizador de vagas especiais.

Este trabalho está organizado da seguinte forma:
- Apresentação inicial da legislação brasileira para as vagas de deficientes físicos e idosos.
- Apresentação de trabalhos e patentes que se assemelham com o trabalho ou são relevantes para o assunto.
- Apresentação dos componentes.
- Explicação da lógica de funcionamento do sistema.
- Detalhamento da construção e programação dos módulos do sistema.
- Resultados dos testes aplicados.
- Conclusão.

2 Objetivos

O presente trabalho propõe a criação de um protótipo simples que fiscaliza e sinaliza as vagas específicas para deficientes físicos e idosos, utilizando duas placas Arduino, dois módulos ESP8266, um sensor de distância HC-SR04 e um conjunto de leitor RFID, demonstrando a sua montagem e função de cada componente. Criação de uma interface no Visual Studio integrada ao Arduino. Testes em bancada, mostrando os resultados do sistema em cima de aplicações simulando a entrada, saída e validação dos veículos pelos usuários.

O objetivo principal desse trabalho seria, futuramente, aplicá-lo em uma vaga existente com o intuito de impedir que essas vagas sejam utilizadas indevidamente por outros usuários, que, consequentemente, prejudicam diretamente os usuários dessas vagas, e em um segundo plano, tentar conscientizar e mudar a cultura brasileira em relação ao uso dessas vagas, tornando o uso correto delas mais visuais através de sinaleiros luminosos.

3 Revisão Bibliográfica

Esta seção visa apresentar as leis, projetos e patentes que contribuíram diretamente na criação do protótipo.

3.1 Legislação

A resolução 304 de 18 de dezembro de 2008 expedida pelo CONTRAN (Conselho Nacional de Trânsito) tem como objetivo uniformizar, em âmbito nacional, os procedimentos para sinalização e fiscalização do uso das vagas regulamentadas, para uso exclusivo de pessoas que transportam portadores de deficiências ou pessoas com dificuldade de locomoção (BRASIL, 2008b).

De acordo com a resolução, há a Lei Federal nº 10.098, de 19 de dezembro de 2000, que em seu artigo 7º, estabelece a obrigatoriedade de ter 2% (dois por cento) das vagas de estacionamento público direcionado a pessoas com deficiência física ou com dificuldade de locomoção, todos devidamente identificados, de acordo com a Figura 1.

Figura 1 – Placa de identificação para vagas destinadas a pessoas que transportam ou são deficientes físicos

Fonte: Geografos, 2012[1]

A identificação dos veículos é feita através de uma credencial, que tem validade em todo o território nacional, sendo emitida pelo órgão ou entidade executiva de trânsito do município de domicílio da pessoa com necessidades especiais. A identificação deve se encontrar em local

[1] Disponível em: http://www.geografos.com.br/placas-de-transito/placa-estacionar-deficiente-fisico.php

visível para efeito de fiscalização. A credencial segue o padrão previsto na resolução conforme a Figura 2.

Figura 2 – Modelo de credencial para deficientes físicos prevista na resolução

Fonte: Deficiente Ciente, 2009[1]

As leis existentes preveem o direito dessas pessoas para a utilização das vagas especiais, porém uma das dificuldades encontradas é que essas leis não se aplicam a espaços privados de caráter público, nos quais shoppings, mercados e outros se encaixam.

O Projeto de Lei nº460/2011 da Câmara Nacional, altera a Lei nº 9.503 de 23 de setembro de 1997 - Código de Trânsito Brasileiro - para promover a fiscalização em edificações que são privadas, mas de uso coletivo (GABRILLI, 2011). Mara Gabrili define em seu Projeto de Lei que edificações privadas de uso coletivo são aquelas de natureza comercial, hoteleira, cultural, esportiva, financeira, turística, recreativa, social, religiosa, educacional, industrial e de saúde.

As leis aplicadas para pessoas idosas são diferentes, no dia 1º de outubro de 2003 foi aprovado o Estatuto do Idoso, Lei nº 10.741, na qual o artigo nº 41 estabelece que seja assegurada a reserva de 5% (cinco por cento) das vagas nos estacionamentos públicos e privados para idosos (BRASIL, 2003).

A Resolução 303/2008 do CONTRAN dispõe sobre as vagas de estacionamento destinadas exclusivamente às pessoas idosas (BRASIL, 2008a), na qual os idosos devem ter os certificados expedidos pelo órgão ou entidade executiva de trânsito do seu município, sendo válido em todo território nacional, seguindo o padrão da Figura 3.

As vagas são devidamente identificadas, e devem ser localizadas em locais de fácil acesso e que necessite pouca locomoção, já que esse grupo de pessoas não possui a mesma condição física de pessoas mais jovens. Esses locais devem ser identificados com placas, de acordo com a Figura 4.

Existe um sistema de identificação, porém não há uma fiscalização precisa que faça com que as vagas sejam usadas exclusivamente por deficientes físicos ou idosos. Este trabalho propõe

[1] Disponível em: https://www.deficienteciente.com.br/contran-e-o-estacionamento-para-idosos_23.html

Figura 3 – Modelo de credencial para idoso prevista na resolução

Fonte: Trânsito e Conhecimento, 2015[1]

Figura 4 – Placas de identificação para vagas destinadas a idosos

Fonte: Trânsito e Conhecimento, 2015[1]

a criação de uma forma de fiscalização mais funcional, para que essas vagas não sejam utilizadas indevidamente.

3.2 Patentes e Projetos

Existe um projeto na Câmara Municipal de Salvador, Projeto de Indicação nº500/15, denominado "Projeto Cidadão Fiscal – Um olho a mais" (PALHINHA, 2015), no qual, há a implantação de aplicativo de celular e site, em que cidadãos previamente cadastrados atuarão como fiscais de desrespeito às vagas de estacionamento destinadas a idosos e deficientes, fazendo o uso dos próprios cidadãos como fiscais, já que não há fiscais suficientes para o serviço.

A patente WO 2011033141 A1, intitulada "System for monitoring car parking spaces reserved for groups of authorized users", apresenta um sistema de monitoramento de vagas para carros autorizados, por meio de um sistema composto por um sensor ultrassônico, que identifica

[1] Disponível em: https://transitoeconhecimento.wordpress.com/2015/05/20/ao-ser-multado-por-estacionamento-em-vaga-de-idoso-o-cidado-reclama-te-contempla-os-agentes-de-trnsito/

a presença de um carro, e de um RFID (Radio Frequency Identification Reader), que é capaz de comunicar com o dispositivo para checar se o veículo estacionado é autorizado. Caso contrário, a polícia é acionada (MOYA; MORTE, 2011).

Apesar de ser um sistema relativamente simples e barato por conta do RFID, o acionamento da polícia pela utilização incorreta de uma vaga é inviável, pois não há a necessidade de acionar a polícia para pequenas infrações.

A patente WO 2007059192 A3, denominada "Permit-based parking environment management method and system", define um sistema que cria um ambiente de estacionamento controlado por um programa de vagas e por etiquetas RFID. Um ou mais RFID são utilizados para escanear os veículos estacionados nas vagas para determinar se a etiqueta está associada ao veículo, permitindo verificar se as vagas estão sendo utilizados de forma correta (LAWRENCE; JOSIAH, 2007). Trata-se de um sistema de baixo custo, porém ainda muito genérico quanto ao funcionamento, pois não apresenta ação quando o veículo estacionado não tem o privilégio de ficar na vaga específica.

A patente US 6501391 B1, com o nome "Internet communication of parking lot occupancy", apresenta um servidor que transmite as informações de uma vaga ocupada por meio da internet, capaz de ser reproduzida em tempo real por um dispositivo eletrônico, podendo fazer uma leitura e retornar as vagas desocupadas de um local (RACUNAS, 2002). A comunicação via internet possibilita ao usuário ter uma informação em tempo real, porém o sistema só tem utilidade para vagas ociosas, não as diferenciando das vagas especiais.

A patente US 7424968 B2, intitulada "Method and apparatus for public street parking using RF and RFID technology", utiliza a tecnologia RFID para monitorar e regular as vagas em ruas, com o objetivo de diferenciar veículos com tíquetes de estacionamento vencidos, veículos estacionados ilegalmente e veículos utilizando tíquetes de outros estacionamentos. A comunicação é feita com uma central que monitora as informações dos veículos estacionados (MEYERHOFER; VICENS, 2008). Não há um objetivo em comum com o projeto proposto neste trabalho, porém a questão da comunicação com uma central é interessante para fazer o monitoramento dessas vagas.

A patente brasileira WO 2013170333 A1, "Sistema de controle de vagas preferenciais em estacionamento", define a criação de um controle de vagas preferenciais em estacionamentos públicos e privados. É um sistema de baixo investimento, baixo custo de operação e manutenção, apresenta facilidade e rapidez na instalação e resistência a vandalismos (YAMAWAKI, 2013). O componente aciona um sistema visual e sonoro, que, de acordo com o autor da patente, a emissão de um alarme sonoro causa constrangimento e motiva a mudar a cultura brasileira quanto a essas vagas preferenciais, no entanto, o sistema só se aplica a veículos que possuem o dispositivo de transmissão para a validação de dados. Portanto, o mesmo poderia causar constrangimento e desconforto à uma pessoa que realmente necessita fazer uso da vaga e não possui o dispositivo.

"A Motorcycle Parking Lot Management System Based of RFID" faz uso do Módulo RFID para a leitura dos cartões *Tags*, linguagem do Visual Basic para a interface com o usuário do sistema e o MySQL para a criação do banco de dados, e essas informações são mostradas em uma página PHP criada pelo sistema. O objetivo do sistema é tornar a busca por vagas e as cobranças dos estacionamentos mais rápidas e eficientes (SHIEH et al., 2013).

O protótipo criado utiliza como base um pouco de cada patente citada, sendo um sistema de fiscalização de vagas especiais fazendo uso da comunicação wifi, validação via RFID, sistema de central para monitoramento e sinalização local da vaga através de leds.

4 Metodologia

4.1 Materiais e Ferramentas

Nesta seção do documento, são apresentados os meios utilizados para a produção do dispositivo de fiscalização de vagas, incluindo os *Hardwares* (componentes físicos) e os *Softwares* (programas) utilizados durante o desenvolvimento do sistema.

4.1.1 Arduino UNO

As placas Arduino são compostas por microcontroladores ATmega328P, que são microprocessadores de 8 bits com 32kB de memória flash (ATMEL, 2016). O Arduino oferece uma plataforma acessível e de fácil programação, na qual pode-se obter inúmeros sensores e componentes de fácil acoplamento (ARDUINO, 2016), como observado na Figura 5.

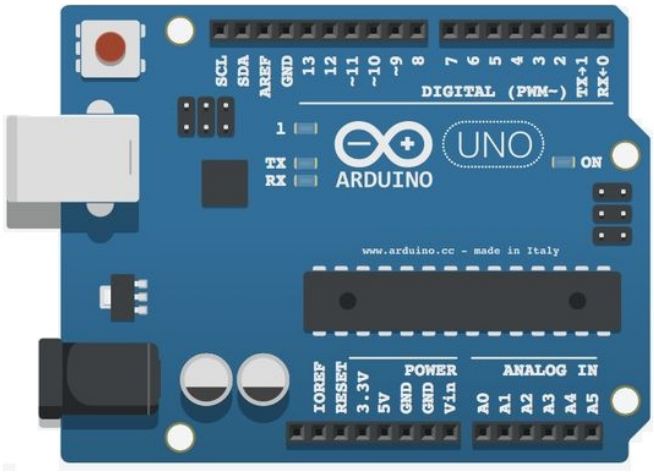


Figura 5 – Placa modelo Arduino UNO

Fonte: Sketch App Resources, 2015[1]

4.1.2 ESP8266

O ESP8266 é um módulo que pode ser acoplado na placa Arduino, o componente recebe as informações via Serial (UART) e interage com a rede via conexões TCP/UDP, permitindo que o microcontrolador consiga se conectar ou até mesmo hospedar uma rede wifi através de sua antena integrada, demonstrado na Figura 6.

[1] Disponível em: https://www.sketchappsources.com/free-source/2077-arduino-uno-board-vector-sketch-freebie-resource.html

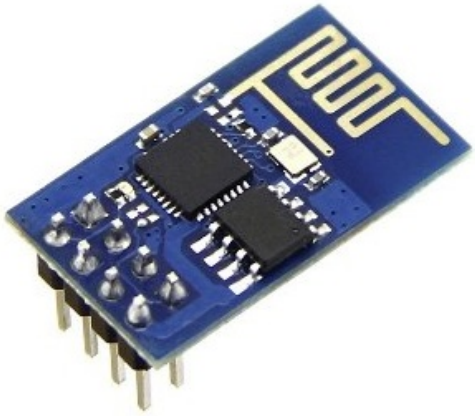


Figura 6 – Módulo Esp8266 modelo ESP-01

Fonte: Filipeflop, 2017[1]

4.1.3 Sensor Ultrassônico

O Sensor Ultrassônico (Figura 7) um componente capaz de calcular a distância de um objeto a sua frente, sendo limitador por um alcance de 2 cm a 400 cm, o seu princípio de funcionamento se baseia em no ultrassom, emitindo uma onda sonora, que é refletida pela superfície do objeto, e, de acordo com o tempo de retorno dessa onda, é calculada a distância (ELECFREAKS, 2013).

Figura 7 – Sensor de Distância Ultrassônico HCSR04

Fonte: Filipeflop, 2016[2]

4.1.4 Módulo RFID

O Leitor RFID (*Radio Frequency Identication*) é um componente que faz a leitura e gravação de *Tags* sem o contato físico (Figura 8), sendo considerado de curto alcance, funcionando

[1] Disponível em: https://www.filipeflop.com/produto/modulo-wifi-esp8266-esp-01/
[2] Disponível em: https://www.filipeflop.com/produto/sensor-de-distancia-ultrassonico-hc-sr04/

através de sinais de rádio operando na frequência de 13,56 Mhz, sendo muito utilizado na substituição dos códigos de barra e em sistemas de segurança que necessitam a validação de dados (NXP, 2016).

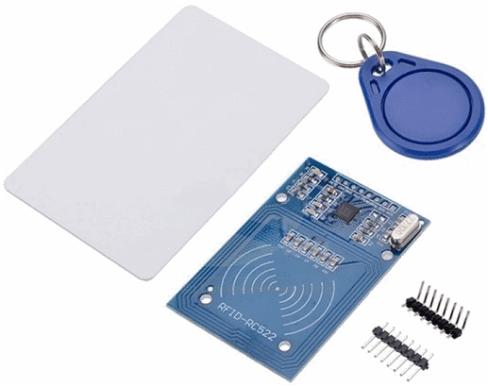


Figura 8 – Conjunto contendo um módulo leitor RFID-RC522, uma *Tag* tipo cartão, uma *Tag* tipo chaveiro, barras de pino lx8 frontal e lx8 lateral

Fonte: Arduino e Cia, 2014[1]

4.1.5 Mb102 - Fonte externa de Energia

O módulo Mb102 criado pela YwRobot cumpre o papel de ser uma fonte de alimentação externa, na qual tem como entrada 12Vcc e transforma a saída em 5v ou 3.3v (Figura 9), na qual sua saída pode ser selecionada por meio de jumpers (YWROBOT, 2014).

Figura 9 – Fonte de alimentacão Mb102 - YwRobot

Fonte: Petervis, 2016[2]

4.1.6 Arduino IDE

O Arduino IDE é a plataforma utilizada para a programação de placas Arduino, capaz de compilar e carregar programas com rapidez e facilidade. Sua programação e feita em C/C++

[1] Disponível em: https://www.arduinoecia.com.br/2014/12/controle-de-acesso-modulo-rfid-rc522.html
[2] Disponível em: https://www.petervis.com/Raspberry_PI/Breadboard_Power_Supply/YwRobot_Breadboard_Power_Supply.html

e é possível interagir com outros dispositivos compatíveis com a placa apenas baixando a sua biblioteca específica (ARDUINO, 2017).

4.1.7 Visual Studio 2017

O Visual Studio é uma ferramenta utilizada para desenvolvimento de softwares, possibilitando a criação de aplicativos móveis, websites, programas para desktops entre outros. Sua linguagem de programação inclui Visual Basic, C, C++, C# e J#, com suporte para Windows, MAC OS, Android e IOS, possibilitando que o desenvolvedor crie uma interface gráfica para o seu aplicativo (VISUALSTUDIO, 2017).

4.2 Funcionamento

Nesta seção é apresentada a arquitetura e programação do projeto. O sistema é composto por dois componentes:

1. *Central*: Componente responsável por receber os sinais do módulo de cada vaga, e, juntamente com um computador, validar ou não a vaga, mostrando em uma interface amigável para o operador.

2. *Módulo Fiscalizador*: Componente responsável por analisar a vaga constantemente e enviar os dados para a central, e de acordo com a resposta da Central, ele indica no sinalizador luminoso específico para cada situação.

Na Figura 10 é possível observar o fluxo de informações do sistema, na qual é utilizada a cor azul para o envio de dados referentes à ocupação da vaga e a cor verde referente a resposta do sistema para a validação da vaga:

1. Os periféricos (Sensor Ultrassônico e o Leitor de Cartões RFID) recebem as informações necessárias (presença de carro e validação da vaga), enviam os respectivos dados ao Módulo Fiscalizador e já acendem o led amarelo indicando que a vaga está ocupada, porém aguardando validação.

2. Por sua vez, o Módulo Fiscalizador recebe essas informações, e as transforma em um código padrão reconhecido pela Central, abre uma conexão com a Central e os enviando via wifi. Caso houver uma validação de *Tag*, ele a envia também e aguarda um posicionamento da Central.

3. A central recebe essa informação, a decodifica e repassa para o computador via USB, que fará a verificação dos dados.

4. A CPU relaciona o código enviado com qual vaga está sendo ocupada, e atualiza a interface gráfica do programa, indicando qual vaga está sendo ocupada, caso houve alguma validação de *Tag*, ela o compara com o banco de dados local, e se for compatível com o mesmo, ela envia uma confirmação de validação para a Central.

5. A Central recebe a verificação do computador via USB, que novamente transmite a resposta para o Módulo Fiscalizador via wifi.

6. O módulo recebe essa informação e de acordo com a resposta recebida, ele acende o led respectivo para a situação.

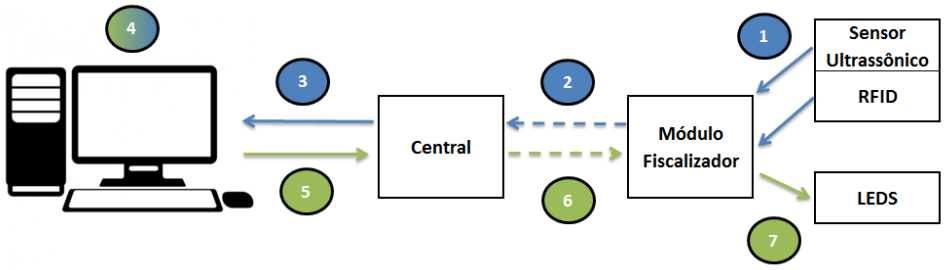


Figura 10 – Fluxo de informações entre o Módulo Fiscalizador e a Central

Fonte: Autor

A lógica de funcionamento pode ser explicada pela Figura 11, na qual existe um loop infinito no fluxograma, fazendo com que o sistema não tenha um fim, possibilitando a verificação da vaga de inúmeros veículos ao longo do tempo.

A sinalização do sistema é demonstrada de duas formas visuais: através de sinaleiros luminosos (leds) para o usuário da vaga e no programa do computador central, demonstrando as informações do usuário para o operador do sistema de fiscalização.

A seleção da cor dos leds sinalizadores seguiu a lógica demonstrada na Tabela 1, na qual foi utilizada a lógica de um semafóro, trocando apenas a cor verde para azul, que é utilizada comumente para sinalizar as vagas especiais.

Tabela 1 – Leds e os estados da vaga

Led	Estado
Vermelho	Vaga Não Validada
Amarelo	Aguardando Validação
Azul	Vaga Validada

Os sinaleiros luminosos só acendem quando há um carro na vaga, o led amarelo indica que a vaga está ocupada, porém aguardando a validação, já o led vermelho é aceso quando o

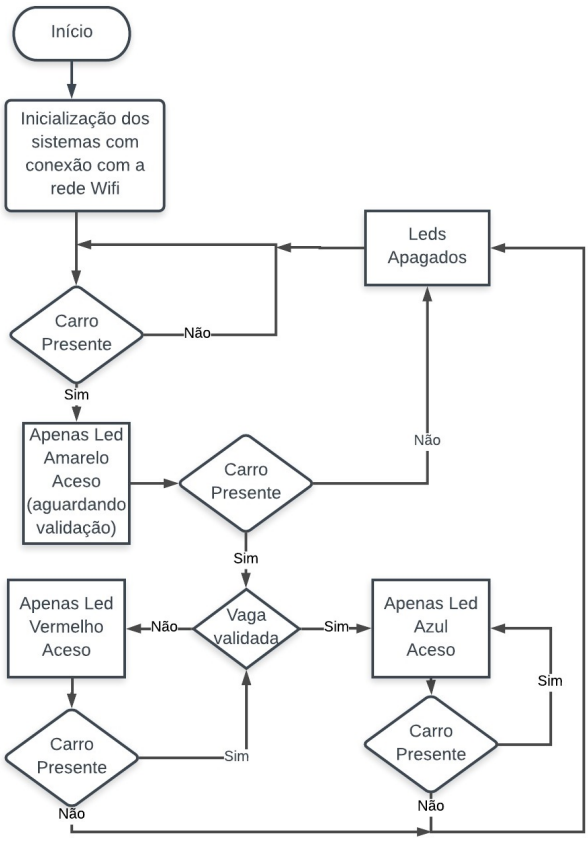


Figura 11 – Fluxograma de funcionamento do sistema

Fonte: Autor

tempo de verificação foi excedido (20 segundos), ou seja, não houve uma validação do usuário na vaga, e o led azul se acende quando o usuário valida a vaga (independente se o tempo de validação foi excedido ou não), conforme demonstrado na Tabela 2.

Tabela 2 – Situações e resposta dos leds

Ocupada	Validação	Led
Não	Não	-
Não	Sim	-
Sim	Não	Amarelo
Sim	Não	Vermelho
Sim	Sim	Azul

Com o intuito de simplificar e melhorar o entendimento do funcionamento de cada componente (Central e Módulo de Fiscalização) as suas seções foram divididas.

4.3 Central

Nesta seção é detalhado o módulo denominado como Central, na qual o seu objetivo é receber as mensagens enviadas pelos módulos e comunicá-los com a CPU central, alterando a interface gráfica do aplicativo do operador.

4.3.1 Montagem

A montagem do sistema central é bem simples, sendo composta por uma placa Arduino UNO, um módulo ESP8266 (ESP-01) e sua via de comunicação com o computador (cabo USB), como indicado na Figura 12.

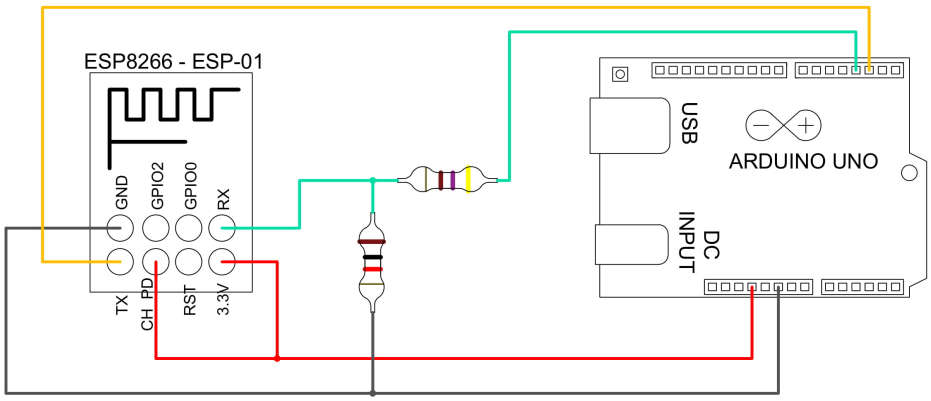


Figura 12 – Diagrama de conexões feitas na Central

Fonte: Autor

O módulo ESP8266 só suporta tensões de 3.3V. Neste projeto, a sua comunicação serial é feita através de pinos digitais (biblioteca *Software Serial* do Arduino), para os pinos Rx e Tx do componente (pinos 3 e 2 respectivamente), já que as portas para comunicação serial 0 e 1 estão sendo utilizadas para a comunicação via USB, então, para que o módulo suporte a tensão de 5V da saída RX, foi aplicado um divisor de tensão, e, utilizando a equação 4.1, foram calculados e inseridos dois resistores, um de 1000Ω e outro de 470Ω, sendo denominados como R2 e R1, respectivamente, resultando em uma saída de aproximadamente 3.4V, que está dentro da faixa aceita pelo módulo, montagem de acordo com a Figura 13.

$$Vout = \frac{R2}{R2 + R1} * Vin = \frac{1000}{1000 + 470} * 5v = \frac{1000}{1470} * 5V = 3.4V \tag{4.1}$$

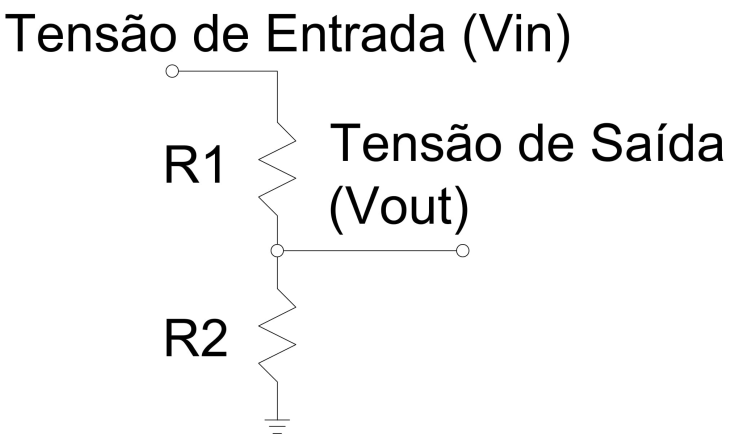


Figura 13 – Esquemático da montagem do divisor de tensão

Fonte: Autor

4.3.2 Programação

4.3.2.1 Arduino

No ambiente do Arduino IDE, foi utilizada a linha de código para a inicialização e configuração do módulo de comunicação wifi para toda vez que ele é iniciado, criando a rede na qual o módulo de fiscalização se conectará (Figura 14), para efeito de conhecimento, toda vez que um módulo ESP8266 hospeda uma rede, ele é denominado como Access Point (AP), ou seja, ponto de acesso. Os comandos utilizados para a sua configuração se encontram no Apêndice A, e o código completo da Central se encontra no Apêndice B.

4.3.3 Visual Studio

A plataforma Visual Studio foi utilizada com dois objetivos:

1. Criar a interface do sistema de fiscalização para o operador
2. Criar e conectar o sistema em um banco de dados na qual será feita a validação da vaga pela *Tag* do cartão RFID

O código completo de programação da Central feita no Visual Studio pode ser encontrada no Apêndice C.

A interface criada foi um layout simples, considerando seis vagas, denominadas A1, A2, A3, B1, B2 e B3, conforme Figura 15.

```
#include <SoftwareSerial.h> //Inclusão da biblioteca de uma porta virtual
SoftwareSerial esp8266(2, 3); //Comunicação serial do ESP8266 com o Arduino nas portas 2 e 3.

void setup()
{
  Serial.begin(9600); //Porta serial do Arduino no Baudrate 9600
  esp8266.begin(9600); //Porta serial virtual do ESP8266 no Baudrate 9600
  sendData("AT+RST\r\n"); //Reset do módulo
  delay(3000) //Delay de 3 segundos
  sendData("AT+CWQAP\r\n"); // Disconecta de qualquer rede que ele esteja conectado
  delay(3000); //Delay de 3 segundos
  sendData("AT+CWMODE=3\r\n"); //Modo de funcionamento como estação e ponto de acesso
  delay(3000); //Delay de 3 segundos
  sendData("AT+CIPMUX=1\r\n"); //Habilitação do modo múltiplas conexões
  delay(3000); //Delay de 3 segundos
  sendData("AT+CIPSERVER=1,333\r\n"); //Criando um servidor na porta 333
  delay(3000); //Delay de 3 segundos
  sendData("AT+CWSAP=\"ESP\",\"1234567890\",5,3\r\n"); //Criando a rede com nome "ESP" e senha "1234567890"
  delay(3000); //Delay de 3 segundos
}
```

Figura 14 – Código de Inicialização da Central

Fonte: Autor

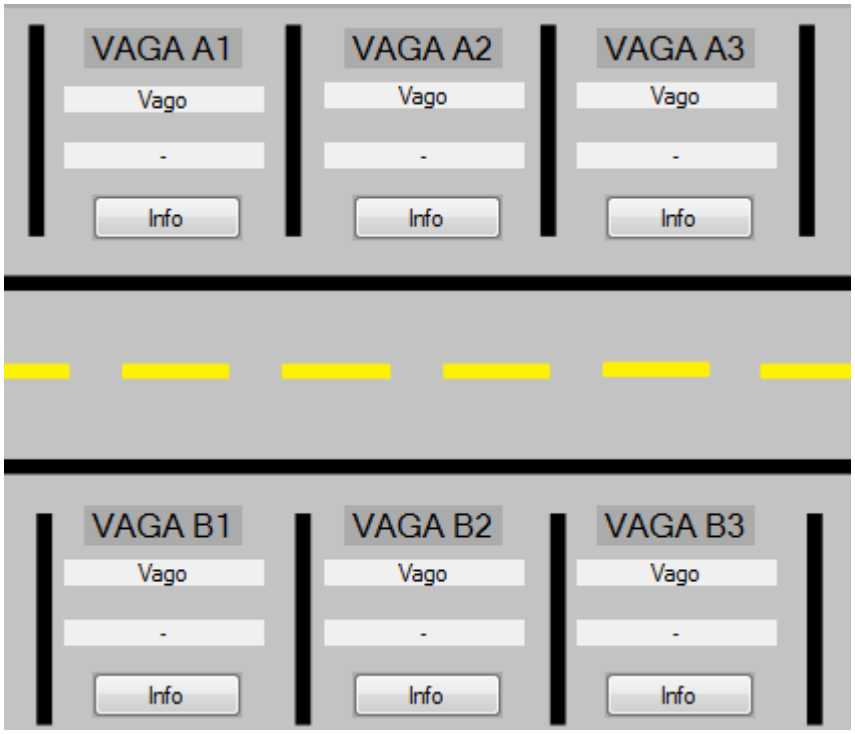


Figura 15 – Layout do programa indicando as vagas

Fonte: Autor

A fim de facilitar a visualização do operador, para cada estado (Vaga Livre, Vaga Ocupada e Vaga Validada), foram utilizadas 3 cores diferentes para cada estado da vaga, conforme demonstrada na Figura 16.

Para entendimento do sistema na identificação de qual vaga está sendo ocupada/validada,

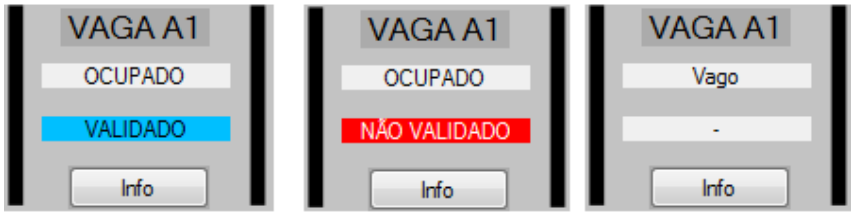


Figura 16 – Possíveis estados de cada vaga

Fonte: Autor

foi determinado um padrão de número a ser enviado pelo Módulo Fiscalizador para a Central, na qual a vaga A1 é definida como número 1, Vaga A2 como número 2 e assim por diante, o código é composto por seis dígitos, sendo eles apresentados na Tabela 3.

Tabela 3 – Código de comunicação para identificação de vaga e *Tag*

Dígito	Código	Variação
1º Dígito	Identificação da Vaga	Variação de 1 a 6, sendo o nº da vaga
2º Dígito	Vaga livre ou Ocupada	0 para livre; 1 para ocupada
3º Dígito	*Tag*	1º Dígito da *Tag* de validação
4º Dígito	*Tag*	2º Dígito da *Tag* de validação
5º Dígito	*Tag*	3º Dígito da *Tag* de validação

Para que o operador consiga visualizar os dados de quem está utilizando o sistema, foi criado um botão para cada vaga (botão "info"), conforme observado na Figura 16, ao clicá-lo, é aberta uma nova janela com as informações completas do usuário, como seu nome completo, a hora de entrada, o documento e o telefone de contato, para caso haja algum tipo de emergência, conforme observado na Figura 17.

Dentro do Visual Studio, essa lógica é feita utilizando variáveis globais, pois as mesmas são intercâmbiadas entre as diferentes "páginas"do programa.

Cada botão é assemelhado um valor de variável, como pode ser observado na Figura 17, essa variável é associada a uma matriz de 6x4, onde as 6 linhas são referentes as 6 diferentes vagas e as 4 colunas são referentes aos dados do usuário.

$$\begin{bmatrix} NomeVaga1 & DataeHorárioVaga1 & DocumentoVaga1 & ContatoVaga1 \\ NomeVaga2 & DataeHorárioVaga2 & DocumentoVaga2 & ContatoVaga2 \\ NomeVaga3 & DataeHorárioVaga3 & DocumentoVaga3 & ContatoVaga3 \\ NomeVaga4 & DataeHorárioVaga4 & DocumentoVaga4 & ContatoVaga4 \\ NomeVaga5 & DataeHorárioVaga5 & DocumentoVaga5 & ContatoVaga5 \\ NomeVaga6 & DataeHorárioVaga6 & DocumentoVaga6 & ContatoVaga6 \end{bmatrix}$$

Nessa matriz são guardadas dinamicamente as informações de cada usuário para cada vaga, e quando o botão é clicado, essa variável muda de localização da matriz, apontando para os dados da vaga selecionada.

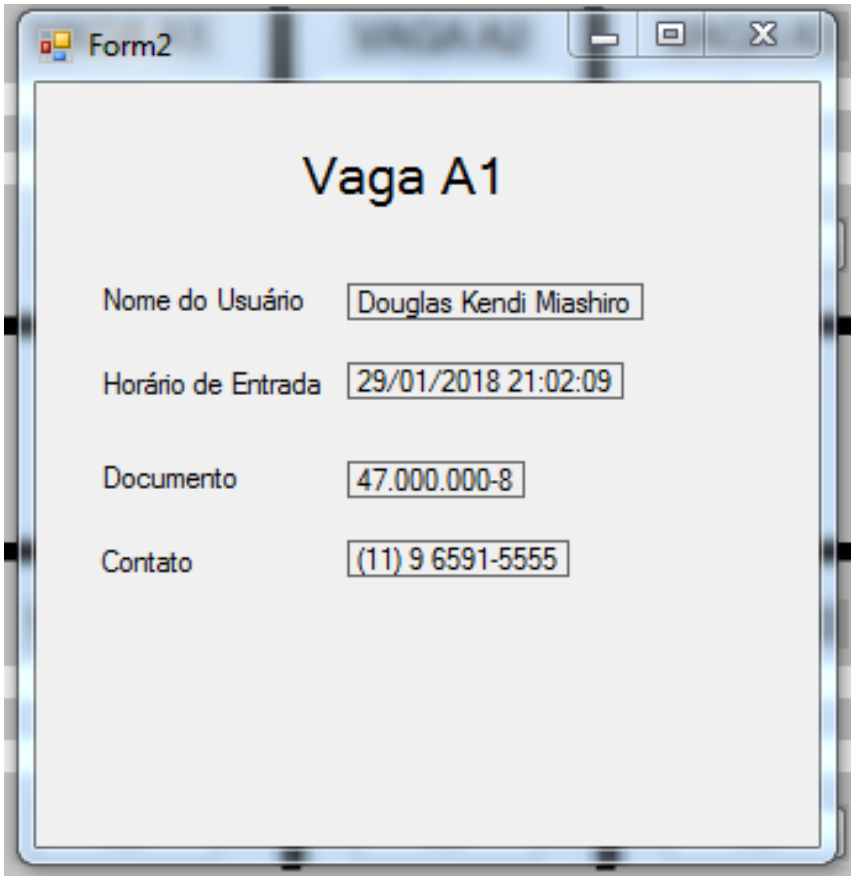


Figura 17 – Informações do usuário presente na Vaga A1

Fonte: Autor

4.4 Módulo Fiscalizador

Nesta seção é apresentado o módulo denominado como Módulo Fiscalizador, na qual seu objetivo é monitorar a vaga com seus periféricos, comunicar com a central e indicar a situação da vaga para o cliente através dos leds.

4.4.1 Montagem

A montagem do módulo fiscalizador se torna um pouco mais complexa, pois há os componentes do campo, que farão o monitoramento e indicação do sistema em relação a vaga, sendo eles:

1. Sensor Ultrassônico (HC SR04): Trabalha como o sensor de presença de carros na vaga, na qual se a distância medida for menor que 1 metro, ele assume que há um carro na vaga.

2. Leitor de Cartões RFID (RFID-RC522): Trabalha como o subsistema de validação de dados, na qual ao passar o cartão *Tag*, ele repassa ao Módulo Fiscalizador o número específico daquele cartão.

3. Módulo de Comunicação (ESP-01): Faz a parte da comunicação, se conectando a rede criada pela Central e enviando os dados da vaga a Central via wifi.

4. Leds: Fazem a sinalização visual da vaga para o usuário do sistema.

5. Módulo de Alimentação Externa (YW Robot 545043): Tem o objetivo de alimentar o módulo ESP-01 e o leitor RFID, já que os mesmos, trabalhando em conjunto, podem chegar a um consumo de 380mA no seu pico, a corrente máxima de consumo da soma das portas do Arduino é de 200mA, os GND do Arduino e do módulo de alimentação devem estar interconectados, para que o referencial terra seja os mesmo para as duas alimentações.

O diagrama de montagem pode ser observado na Figura 18.

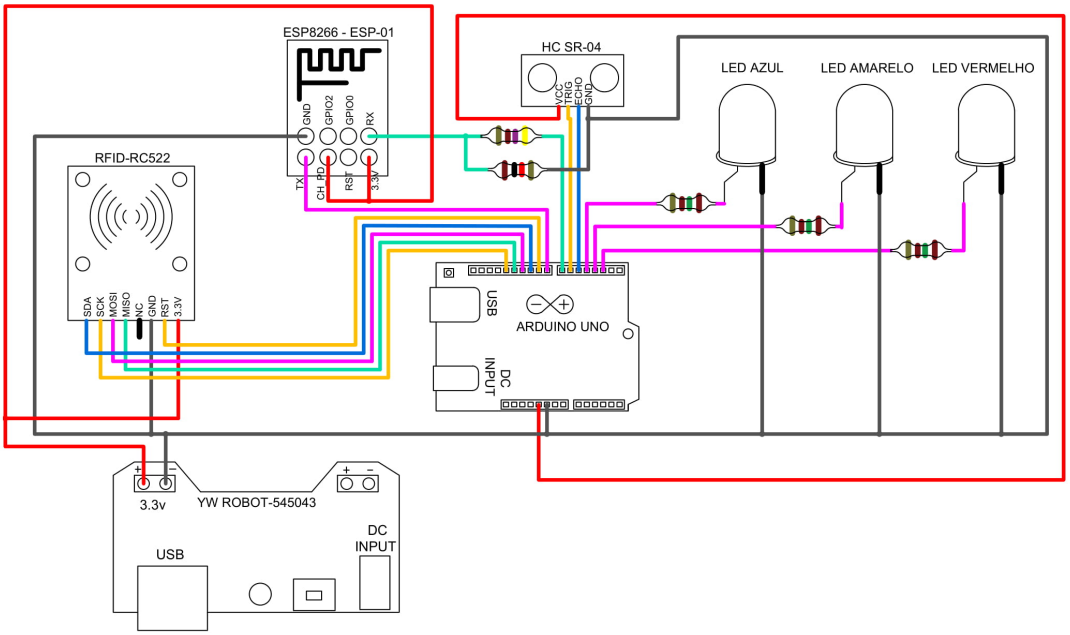


Figura 18 – Diagrama de conexões feitas no Módulo Fiscalizador

Fonte: Autor

Novamente, para a comunicação do pino RX (conexão com o pino 7 do Arduino) do ESP8266, foi aplicado um divisor de tensão com o auxílio de dois resistores de 1kΩ e de 470Ω para que a tensão recebida seja de 3.3V.

Para cada um dos leds, foi utilizado um resistor em série de 150Ω em seu anodo, de acordo com a Figura 19, essa montagem deve ser feita para limitar a corrente que atravessará o led, utilizando a Equação 4.2, e com os datasheets dos leds, se tem a informação de que a sua queda de tensão é de 2v e consumo de 20mA(KINGBRIGHT, 2015), considerando a saída de 5v da saída digital do arduino.

$$R = \frac{(Varduino - Vled)}{Iled} = \frac{5v - 2v}{20mA} = \frac{3v}{0,02A} = 150\Omega \tag{4.2}$$

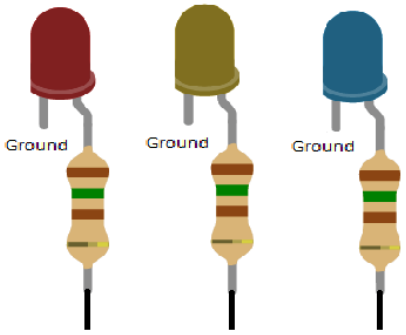


Figura 19 – Conexões dos 3 leds no sistema com um resistor de 150Ω no Anodo de cada Led

Fonte: Autor

4.4.2 Programação

4.4.2.1 Arduino

No Arduino IDE foi feita a programação para resposta do sistema em relação a interação do usuário com a vaga, o programa e comentado pode ser observado no apêndice D. Na configuração do ESP8266 do programa, é feita a conexão do módulo fiscalizador com a central, conforme pode ser observado na Figura 20.

Toda vez que um carro para na vaga especial, o led amarelo se acende, indicando que vaga está ocupada sem validação, porém, para o sistema, ele entende que a vaga se encaixa no estado "Vaga Ocupada e sem Validação", já que não houve nenhuma verificação.

Como o sistema não pode aguardar uma validação ou o tempo de verificação, ele envia um código indicando que a vaga foi ocupada, independente se haverá validação ou não por parte do usuário. Para isso, o Módulo Fiscalizador abre uma conexão com a Central utilizando o comando *AT+CIPSTART*, na qual ele pode enviar mensagens utilizando o comando *AT+CIPSEND*, assim, o módulo envia o código utilizando como base a Tabela 4 (Numeração definida entre as vagas):

```
Serial.begin(9600); //Porta serial do Arduino no Baudrate 9600
esp8266.begin(9600); //Porta serial virtual do ESP8266 no Baudrate 9600
sendData("AT+RST\r\n"); //Reset do módulo
delay(3000) //Delay de 3 segundos
sendData("AT+CWMODE=1\r\n"); //Modo de funcionamento como estação
delay(3000);//Delay de 3 segundos
sendData("AT+CIPMUX=1\r\n"); //Habilitação do modo múltiplas conexões
delay(3000);//Delay de 3 segundos
sendData("AT+CIPSERVER=1,333\r\n"); //Criando um servidor na porta 333
delay(6000);//Delay de 3 segundos
sendData("AT+CWJAP=\"ESP\",\"1234567890\"\r\n"); //Conectando a rede com nome "ESP" e senha "1234567890"
delay(3000);//Delay de 6 segundos
```

Figura 20 – Código de inicialização do Módulo Fiscalizador

Fonte: Autor

Tabela 4 – Código de envio para vaga está ocupada

Dígito	Código
1º Dígito	Identificação da Vaga
2º Dígito	1 (Ocupada)
3º Dígito	0
4º Dígito	0
5º Dígito	0

O módulo envia os campos referentes a *Tag* como "0000"já que no banco de dados, essa *Tag* foi reservada para a comunicação indicando vaga ocupada sem validação. Após o envio da mensagem, o módulo fecha a conexão utilizando o comando *AT+CIPCLOSE*, caso o usuário validar a vaga com o cartão, o sistema abre novamente uma conexão com a central, envia a numeração de acordo com a *Tag* lida e aguarda a resposta da central em relação a validação do número da *Tag* com o banco de dados. Se a resposta for positiva, o sistema altera o led aceso para azul, caso a resposta for negativa ou o tempo de validação for excedido sem validação, o sistema altera o led aceso para vermelho, vale ressaltar que a validação do sistema pode acontecer mesmo após o tempo de verificação do sistema.

Para o Módulo Fiscalizador entender se a vaga foi validada, ele verifica se a mensagem recebida foi *"+IDP,0,2:OK"*, ele acende o led azul, indicando que a tag testada foi validada pelo sistema, caso contrário (receber a mensagem *"+IDP,0,2:NO"* ou não receber resposta nenhum do sistema depois de 10 segundos) ele fecha a conexão e se mantém no estado atual.

5 Resultados e Discussão

Nesta seção são apresentados e discutidos os resultados obtidos na confecção do protótipo.

Em um primeiro momento foi feita a conexão entre os módulos, sendo criado um AP (access point) para a hospedagem da rede (Figura 21), sendo denominada como "ESP", com senha "1234567890", canal 1 e chave de encriptação do tipo WPA2_PSK.

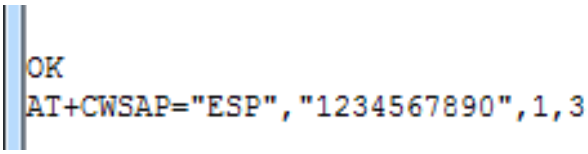


Figura 21 – Criação da rede (ponto de acesso)

Fonte: Autor

Tendo a rede estabelecida, foi possível conectar o outro módulo, utilizando como dados o nome da rede e sua respectiva senha, conforme indicado na Figura 22.

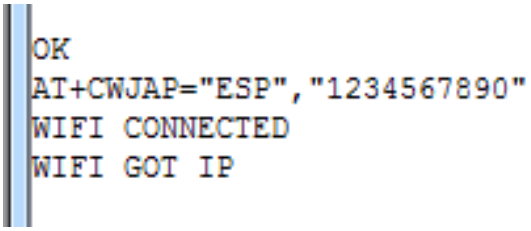


Figura 22 – Conexão da estação no ponto de acesso

Fonte: Autor

É possível fazer a checagem de quais dispositivos estão conectados e quais são os seus repectivos IP's na rede através do comando *AT+CWLIF* (Figura 23).

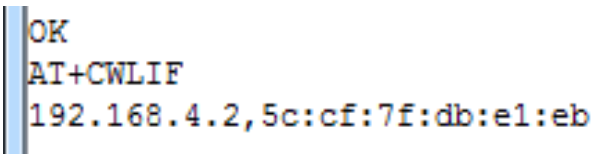


Figura 23 – Execução do comando AT+CWLIF, indicando quais dispositivos e seu respectivo IP conectado a rede criada

Fonte: Autor

Para testar o intercâmbio de mensagens entre os módulos, devem ser utilizados os comandos:

1. *AT+CIPSTART*: responsável por abrir uma conexão dedicada entre a estação e o ponto de acesso.

2. *AT+CIPSEND*: responsável por detalhar a mensagem a ser enviada de um módulo para o outro.

Conforme a Figura 24, é observado o monitor serial da Estação.

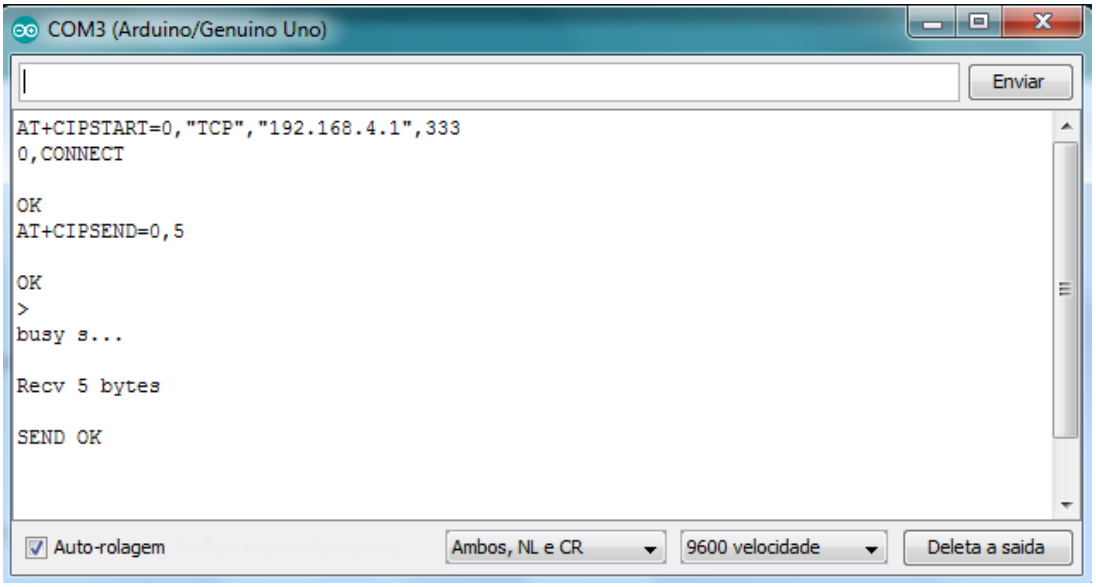


Figura 24 – Imagem do Monitor Serial da Estação para a conexão e troca de mensagens entre módulos

Fonte: Autor

Para o teste, foram informados os dados para o comando: 0 (id do canal de comunicação); TCP (Protocolo de transmissão que ordena a mensagem e garante que será entregue sem que sua ordem seja afetada); 192.168.4.1 (Indicação do ip do ponto de acesso na qual será feita a comunicação); 333 (Indicação da porta na qual será feita a comunicação).

Após o envio do comando, a mensagem *0, CONNECT* aparece, significando que o link entre os módulos foi feito. Essa mesma mensagem é observada no monitor serial do ponto de acesso (Figura 24).

Para o envio da mensagem, é utilizado o comando *AT+CIPSEND*, na qual os seus parâmetros são o id e o tamanho da mensagem a ser enviada, respectivamente, nesse caso foi testada o envio da mensagem "TESTE".

Na Figura 25, é possível verificar o monitor serial do Ponto de Acesso.

Um ponto relevante seria que o ponto de acesso deve estar configurado para aceitar múltiplas conexões (comando *AT+CIPMUX=1*) e com um servidor criado e com a porta definida (no nosso caso, é utilizada a porta 333 definida pelo comando *AT+CIPSERVER=1,333*).

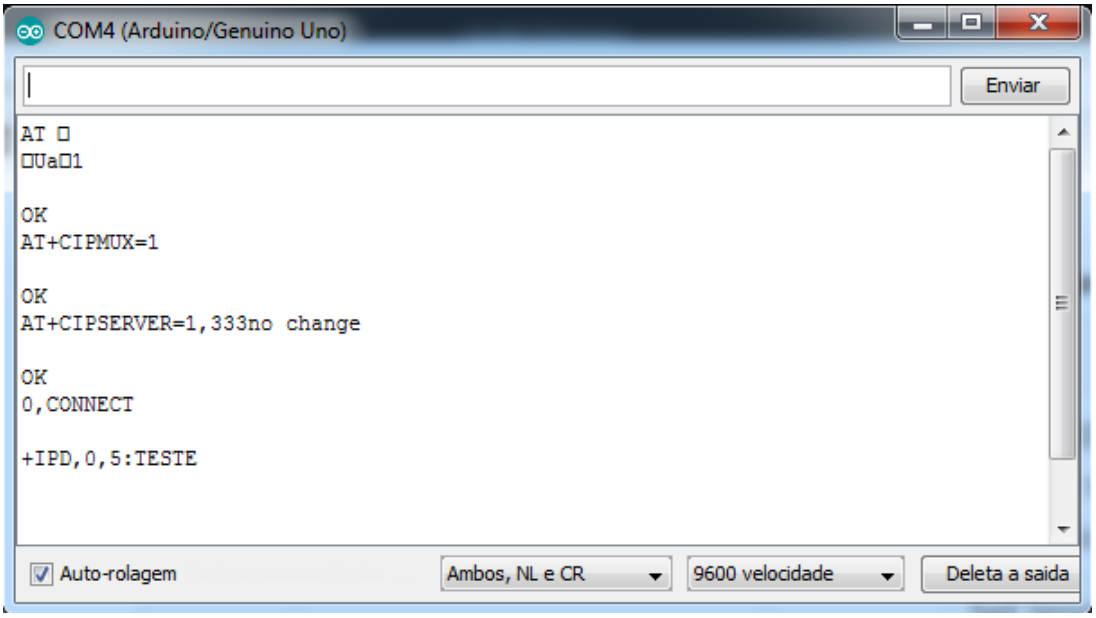


Figura 25 – Imagem do Monitor Serial do Ponto de Acesso para a conexão e troca de mensagens entre módulos

Fonte: Autor

A linha *0, CONNECT* acontece devido ao comando *AT+CIPSTART* feito pelo outro módulo, e na linha posterior, é observado que a mensagem enviada pelo outro módulo (TESTE) foi recebida no formato *+IPD,0,5:TESTE*, na qual o parâmetro 0 é o id da comunicação feita e 5 seria a quantidade de caracteres da mensagem.

Após realizar alguns testes contínuos de troca de mensagens entre os módulos, foi possível verificar que o seu *delay* entre o envio e recebimento é muito pequeno, sendo menor que 1 segundo quando não há nenhum problema no seu envio, para 30 mensagens enviadas, apenas 1 não foi enviada, deixando o componente ESP8266 em modo *busy* (ocupado), na qual teve que ser reiniciado para voltar a conexão. Não foi possível fazer a medição do tempo exato de atraso entre o envio e o recebimento da mensagem devido a falta de equipamentos e softwares específicos.

Em um segundo momento, foi testado a integração do Arduino com o Visual Studio, verificando se as mensagens enviadas pelos módulos podem ser repassados ao computador via cabo USB.

Para a bateria de testes, foram criados alguns usuários testes no banco de dados dentro do Visual Studio com os campos Nome, *Tag*, Deficiente ou Idoso, Documento e Telefone, conforme demonstra a Figura 26.

O campo *Tag* é o valor único de cada cartão RFID com o seu respectivo usuário. Com o intuito de diferenciar os dois tipos de usuários (deficientes físicos e idosos), foram utilizados dois números diferentes, sendo 0 para usuários com deficiência física e 1 para usuários idosos.

Foi criada também um layout para testes (Figura 27), na qual existem itens para verificar

	Id	Nome	Tag	Def ou Idoso	Documento	Telefone
	1	Douglas Kendi ...	192	1	47.050.259-8	(11) 9 6591-5555
	2	Tayná Calaes	105	0	MG-xx.xxx.xxx	(31) 9 9999-1815
	3	Sueli Miashiro	134	1	00.000.000-0	(11) 9 1234-5678
▷	NULL	NULL	NULL	NULL	NULL	NULL

Figura 26 – Dados de usuários fictícios criados para os testes

Fonte: Autor

se a conexão está sendo feita corretamente e se as mensagens recebidas estão sendo filtradas de forma correta.

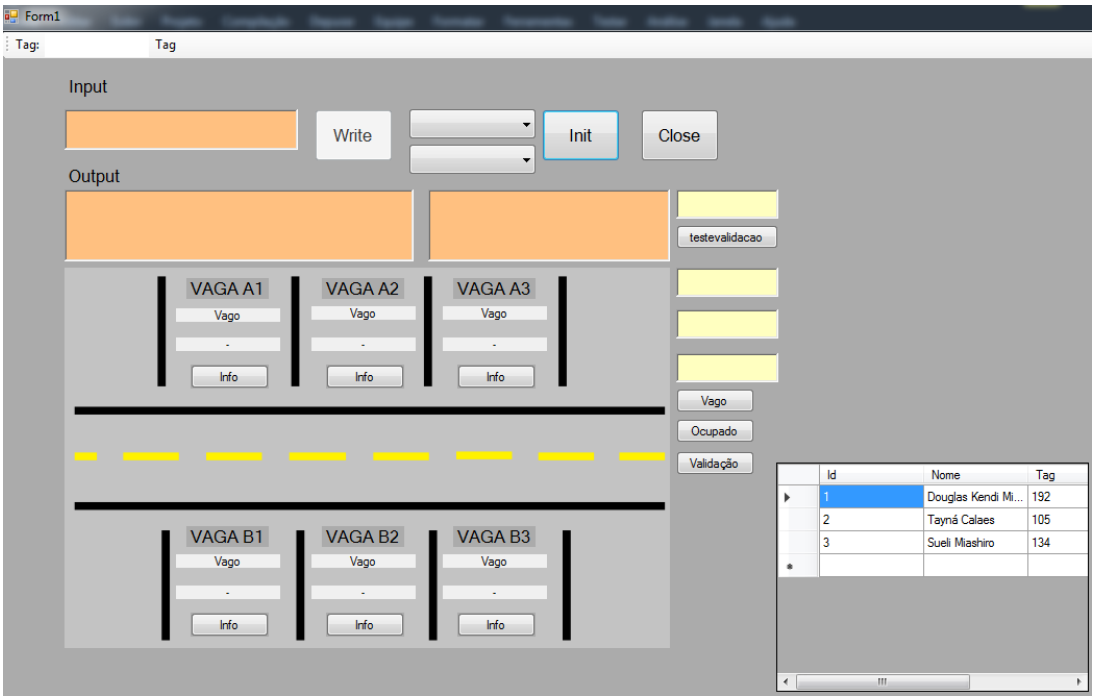


Figura 27 – Layout de testes criada para verificação da comunicação entre o programa e a porta serial

Fonte: Autor

Para iniciar a conexão entre o programa do Visual Studio e a porta serial de recebimento, foram criadas caixas de seleção indicando em qual porta será feito a comunicação (Figura 28) e qual a velocidade da taxa de transmissão dessa comunicação (Figura 29).

Na configuração da caixa de seleção de portas, ela filtra e deixa como opção as portas disponíveis na qual o Arduino está conectado, neste caso, seria a porta COM4.

Para a seleção do *Baud Rate* (taxa de transmissão), há as opções de 9600 bits/s, 38400 bits/s, 57600 bits/s e 115200 bits/s, que são as taxas de transmissões mais comuns utilizadas. Para o programa, é selecionada a velocidade 9600 bits/s, já que o componente ESP8266 utiliza a

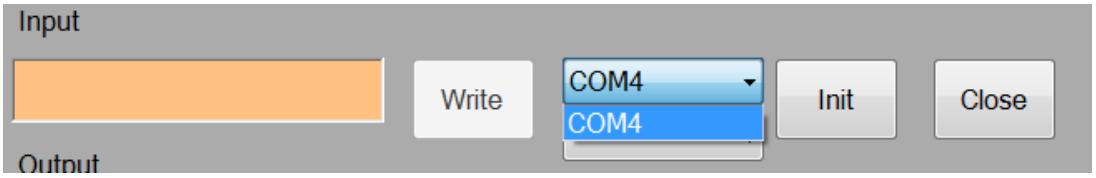


Figura 28 – Tela de testes para seleção da porta de comunicação entre o programa e a porta serial

Fonte: Autor

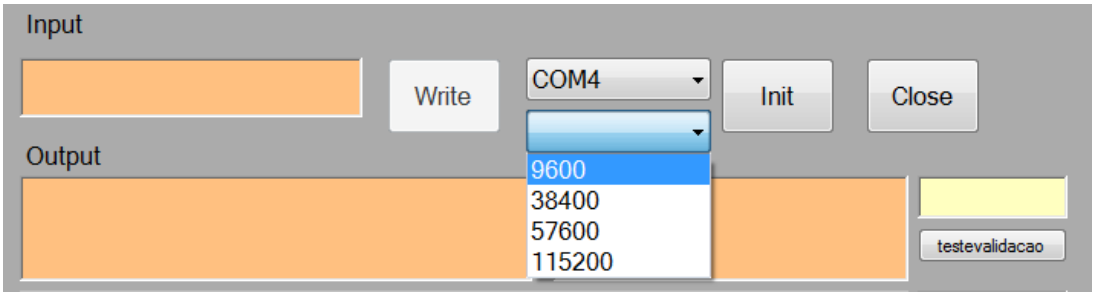


Figura 29 – Tela de testes para seleção da velocidade (Baud Rate) da taxa de transmissão entre o programa e a porta serial

Fonte: Autor

mesma velocidade para se comunicar com a porta serial.

No layout de testes (Figura 27), existem os campos *Input* e *Output*, no qual tem a mesma função do monitor serial do Arduino, que seria o envio e recebimento de comandos e mensagens do Arduino IDE.

Para testar se os campos estavam sendo preenchidos com as mensagens enviadas e recebidas pela porta serial, foi rodado o comando *AT*, na qual deve ter como resposta do ESP8266 a mensagem *OK*, como pode ser observado na Figura 30.

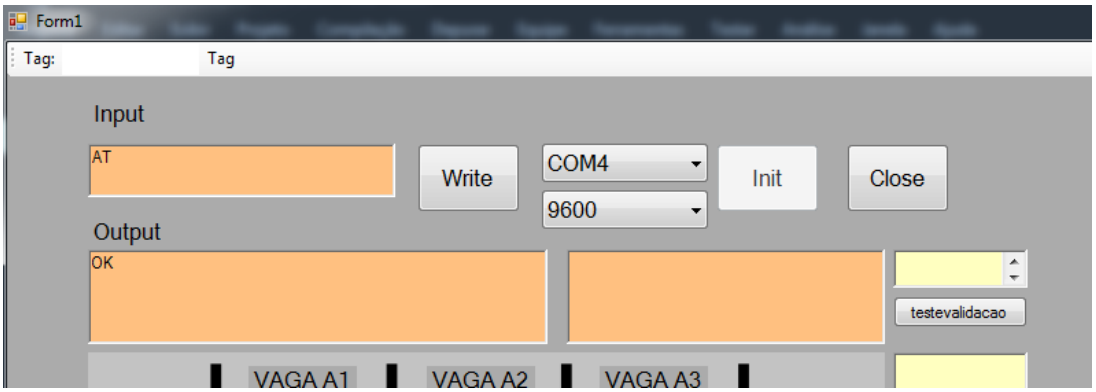


Figura 30 – Tela de testes para verificação da comunicação entre o programa e a porta serial

Fonte: Autor

Para testar se a programação do aplicativo estava funcionando corretamente, foi enviada a mensagem do Módulo Fiscalizador para a Central simulando a utilização da vaga monitorada (Figura 31)

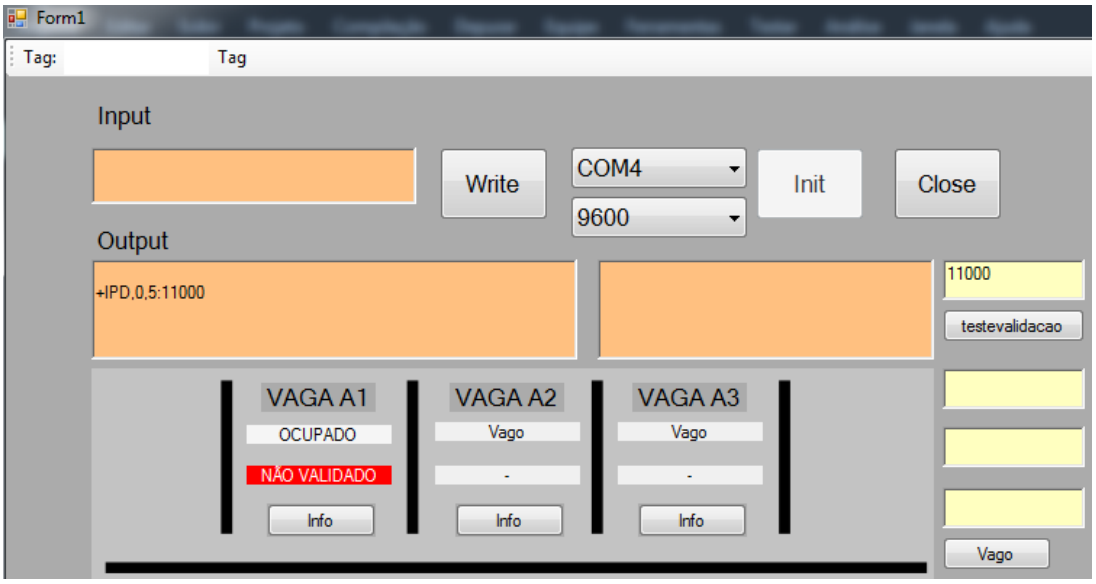


Figura 31 – Tela de testes para simulação de ocupação da vaga A1

Fonte: Autor

Para que o programa criado no Visual Studio consiga processar o código definido (Tabela 3), ele utiliza uma de suas funções denominada *Substring(X,Y)*, no qual seu objetivo é separar a *string* recebida pelo módulo em um código aceito, desconsiderando o prefixo "*+IPD,0,5:*". As variáveis X e Y da função são trocadas pelo caractere inicial em que se deve começar a contagem e a quantidade de caracteres a ser considerados de acordo com o início da contagem, respectivamente. Para o teste, é observado que no campo da direita, essa função está sendo aplicada, filtrando a mensagem em apenas "11000", que consequentemente já indica visualmente que a vaga A1 foi ocupada.

Após o teste da vaga ocupada, foi feita uma simulação baseada no banco de dados (Figura 26), na qual o usuário Douglas Kendi Miashiro valida a vaga com o seu respectivo cartão RFID, resultando na Figura 32.

Após a simulação, foi feita uma bateria de testes na bancada: a presença do carro foi simulada com um objeto apenas para indicar a presença do veículo; e foi utilizado o banco de dados com clientes fictícios, já alinhados com a numeração dos 3 cartões RFID disponíveis.

A Tabela 5 indica os resultados da bateria de testes aplicados:

Para os testes, os módulos ficaram energizados em três períodos distintos:
-26/01/2018: de 10:45 às 21:32
-27/01/2018: de 08:12 às 23:19

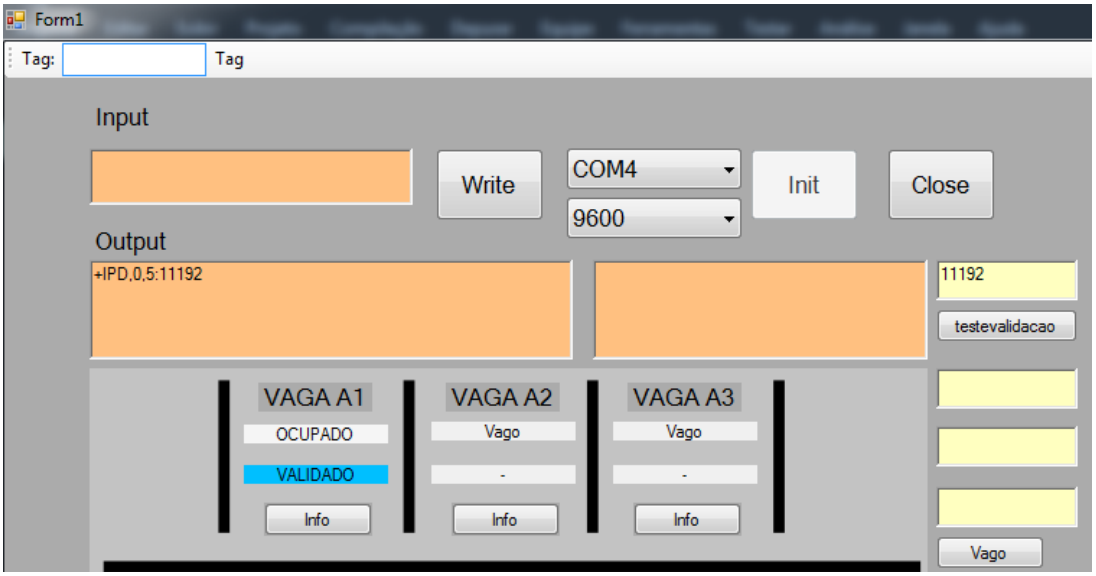


Figura 32 – Tela de testes para simulação de validação da vaga A1

Fonte: Autor

-04/02/2018: de 11:21 às 23:45

Durante esses períodos os componentes do sistema ficaram ligados, sendo apenas reiniciados quando um problema era encontrado.

Para a Tabela 5, foram encontrados os seguintes erros:

1. ERRO A - Variação da distância verificada pelo sensor ultrassônico: em alguns momentos, o sensor mede alguns picos de distâncias, que fazem com que o sistema reinicie, como se o carro atual tivesse saído e um outro carro entrado na vaga, fazendo com que a validação do carro atual seja perdida.

2. ERRO B - Envio simultâneo de mensagens: quando as mensagens enviadas pelo comando *AT+CIPSEND* ocorrem em um curto espaço de tempo (em menos de 3 segundos), as mensagens não conseguem ser enviadas, pois o módulo entra em modo *busy* (ocupado), devido a necessidade de confirmar o envio e recebimento de cada mensagem, fazendo com que o sistema não funcione corretamente.

3. ERRO C - Incialização do módulo: quando o ESP8266 é ligado/reiniciado ele começa a configuração/conexão das redes, em alguns casos, ele não inicia corretamente e consequentemente os seus comandos não são executados. Até o momento não foi encontrada a causa desse problema.

4. ERRO D - Mensagens recebidas parciais: a mensagem chegou de forma parcial, pois no momento em que o código foi lido pelo programa, ele chegou de forma parcial, ocorreu 1 vez nos 25 testes aplicados, o problema pode ser devido a forma de leitura entre o Arduino

Tabela 5 – Tabela de testes aplicados

Teste nº	Data	Entrada	Saída	Validação	Horário Verif.	Resultados
1	26/01/2018	10:53	11:00	Não	-	OK
2	26/01/2018	11:03	11:20	Sim	11:03	OK
3	26/01/2018	11:21	12:20	Sim	11:27	OK
4	26/01/2018	15:00	15:15	Não	-	OK
5	26/01/2018	15:15	17:00	Não	-	OK
6	26/01/2018	17:30	19:43	Sim	18:00	ERRO A
7	26/01/2018	19:44	20:14	Sim	20:01	ERRO D
8	26/01/2018	20:22	21:20	Sim	20:23	OK
9	26/01/2018	21:24	21:27	Não	-	OK
10	27/01/2018	08:17	08:59	Não	-	OK
11	27/01/2018	09:00	10:49	Sim	09:13	OK
12	27/01/2018	11:04	11:04	Sim	11:04	OK
13	27/01/2018	11:50	13:02	Sim	13:00	OK
14	27/01/2018	13:03	14:00	Não	-	ERRO B
15	27/01/2018	19:08	20:00	Não	-	OK
16	27/01/2018	20:01	20:12	Não	-	OK
17	27/01/2018	20:13	20:19	Sim	20:13	OK
18	27/01/2018	20:37	21:29	Sim	20:46	OK
19	27/01/2018	20:47	21:02	Não	-	OK
20	27/01/2018	21:05	21:32	Não	-	OK
21	27/01/2018	21:42	23:12	Sim	21:43	OK
22	03/02/2018	12:15	17:12	Não	-	ERRO C
23	03/02/2018	17:12	20:00	Sim	17:12	OK
24	03/02/2018	20:21	20:37	Não	-	OK
25	03/02/2018	22:00	23:44	Sim	23:01	OK

e o componente ESP8266, que em sua programação, considerou a cadeia de caracteres recebidos como a *string* "response", sem fazer uma verificação da contagem de caracteres.

5. ERRO E - Formato do recebimento de mensagens: nem sempre as mensagens recebidas vem no mesmo formato, em algumas vezes ela vem na mesma linha de um comando anterior, para o Visual Studio não houve consequências, pois o mesmo consegue fazer a leitura do código mesmo com o espaçamento diferente entre as mensagens recebidas, por essa razão ele não foi indicado na tabela.

6 Considerações Finais

6.1 Conclusão

Com esse trabalho foi possível criar um protótipo de dispositivo simples e de baixo custo, utilizando duas placas Arduino, dois módulos de comunicação wifi ESP8266, um conjunto de leitor RFID, um sensor de distância HC-SR04 e a plataforma Visual Studio para integrar os dados da comunicação em uma interface para o operador do sistema.

Conforme os resultados demonstrados, foi possível fazer a conexão entre os módulos wifi ESP8266 (Central e Módulo Fiscalizador), comunicando o estado atual da vaga através de estruturas de códigos criadas; criar e integrar um banco de dados de usuários, contendo os dados a serem demonstrados na interface do programa; integrar a interface do programa criado no Visual Studio com o Arduino via cabo USB e testar o protótipo em bancada.

O módulo ESP8266 apesar de ser um componente de baixo custo, versátil e que possibilita a criação e conexão do mesmo em uma rede wifi, é um módulo delicado e sem muita informação para utilizar como base. Algumas características foram observadas: suas conexões não devem estar com mau contato ou a comunicação estará debilitada; os seus pinos são de 3.3v e seu consumo pode ultrapassar o limite ofertado pela placa Arduino, caso contrário ele será danificado ou não funcionará corretamente e a sua versão de firmware pode ser atualizada, mas apenas por usuários que tenham conhecimento profundo sobre o módulo, pois em inúmeros casos relatados pelos usuários, o módulo foi inutilizado.

O sensor de distância HC-SR04 é um componente simples e de fácil integração com o Arduino, visto que existem inúmeros projetos e bibiotecas disponíveis na rede, porém sua aplicação no protótipo trouxeram algumas variações de medições de distâncias inesperados, comprometendo diretamente o funcionamento do protótipo (verificação de veículo presente ou não), reiniciando o sistema e descartando qualquer validação feita pelo usuário.

Apesar dos problemas encontrados, os resultados dos testes de funcionamento do protótipo foram satisfatórios (apenas 4 dentre os 30 testes feitos foram não alcançaram o objetivo), pois foi possível, em uma bancada, monitorar o estado de uma vaga criada, simulando a entrada, saída e validação da vaga, testando quais as possíveis falhas do sistema. Vale ressaltar que melhorias devem ser feitas e novos testes devem ser aplicados para que, futuramente, o sistema possa ser implantado em uma vaga real.

6.2 Trabalhos Futuros

Para trabalhos futuros sugere-se alguns pontos a serem trabalhados:

1. Aplicação de um filtro para o sinal recebido do sensor ultrassônico.
2. Criar um pequeno *delay* entre o envio e recebimentos de mensagens, possibilitando que o componente ESP8266 consiga enviar e receber as mensagens sem entrar no modo *busy*.
3. Criar uma forma de contar os caracteres recebidos em cada mensagem, fazendo com que essa função funcione como um verificador de mensagens, e, em caso negativo, o sistema requisitaria um novo envio da mensagem.
4. Criar um banco de dados de interações das vagas com o usuário, obtendo um histórico das vagas.
5. Fazer um estudo aprofundado em relação ao ESP8266, entendendo o motivo de suas mensagens chegarem em formatos diferentes e também do seu problema de inicialização já citado.
6. Implantação de um sistema de cadastro na Central, criando uma interface para criação e alteração de cadastros do sistema, sendo necessário a utilização de mais um componente Leitor RFID RC-522 para que o cadastro seja dinâmico.
7. Criação de uma proteção a prova de intempéries para o módulo fiscalizador, já que nesse trabalho não foi possível rodar os teste em um ambiente externo.
8. Criar mais módulos fiscalizadores e testar o sistema com mais de um módulo, a fim de delimitar uma relação de quantidades de módulos fiscalizadores para uma central.
9. Prever uma infraestrutura simples e de fácil implementação do sistema em uma vaga existente.

Referências

ARDUINO. *Arduino Uno Datasheet*. 2016. Disponível em: https://www.farnell.com/datasheets/1682209.pdf. Acessado em 25/02/2017. 8

ARDUINO. *Arduino IDE 1.8.5*. 2017. Disponível em: https://www.arduino.cc/en/Main/Software. Acessado em 25/02/2017. 11

ATMEL. *ATmega328/P Datasheet*. 2016. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. Acessado em 27/02/2017. 8

BRASIL. *Lei Nº 10.741, de 1º de Outubro de 2003, Dispõe sobre o Estatuto do Idoso e dá outras providências*. 2003. 4

BRASIL. *Resolução 304 de 18 de dezembro de 2008, Dispõe sobre as vagas de estacionamento de veículos destinadas exclusivamente às pessoas idosas*. 2008. Conselho Nacional de Trânsito, CONTRAN, 2008. 4

BRASIL. *Resolução 304 de 18 de dezembro de 2008, Dispõe sobre as vagas de estacionamento destinada exclusivamente a veículos que transportem pessoas portadoras de deficiência e com dificuldade de locomoção*. 2008. Conselho Nacional de Trânsito, CONTRAN, 2008. 3

ELECFREAKS. *Ultrasonic Ranging Module HC-SR04 Datasheet*. 2013. Disponível em: http://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf. Acessado em 07/06/2017. 9

GABRILLI, M. *PL 460/2011*. 2011. Altera a lei Nº 9.503 de 23 de Setembro de 1997 – Código de Trânsito Brasileiro – para promover a fiscalização em edificações privadas de uso coletivo e dá outras providências. 4

KINGBRIGHT. *Leds Datasheet*. 2015. Disponível em: https://cdn-shop.adafruit.com/datasheets/WP7113SRD-D.pdf. Acessado em 25/02/2017. 20

LAWRENCE, B.; JOSIAH, J. *Permit-based parking environment management method and system*. [S.l.]: Google Patents, 2007. WO Patent App. PCT/US2006/044,300. 6

MEYERHOFER, E.; VICENS, E. *Method and apparatus for public street parking using RF and RFID technology*. [S.l.]: Google Patents, 2008. US Patent 7,424,968. 6

MOYA, B.; MORTE, C. *System for monitoring car parking spaces reserved for groups of authorised users*. [S.l.]: Google Patents, 2011. WO Patent App. PCT/ES2009/000,470. 6

NXP. *MFRC522 Standard performance MIFARE and NTAG frontend*. 2016. Disponível em: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf. Acessado em 14/10/2017. 10

PALHINHA, O. *"Projeto Fiscal Cidadão – Um olho a mais*. 2015. Câmara Municipal de Salvador, 67ª Sessão Ordinária, 3º Período Legislativo, 17ª Legislatura. 5

RACUNAS, R. *Internet communication of parking lot occupancy*. [S.l.]: Google Patents, 2002. US Patent 6,501,391. 6

SHIEH, H. L. et al. A motorcycle parking lot management system based on rfid. In: *2013 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*. [S.l.: s.n.], 2013. p. 268–272. ISSN 2377-5823. 7

VISUALSTUDIO. *Visual Studio Release Notes*. 2017. Disponível em: https://www.visualstudio.com/pt-br/news/releasenotes/vs2017-relnotes. Acessado em 25/02/2017. 11

YAMAWAKI, S. *Sistema de controle de vagas preferenciais em estacionamento*. [S.l.]: Google Patents, 2013. WO Patent App. PCT/BR2013/000,174. 6

YWROBOT. *Breadboard Power Suplly Datasheet*. 2014. Disponível em: https://hobbyking.com/media/file/403178644X1017066X57.pdf. Acessado em 15/04/2017. 10

Apêndices

APÊNDICE A – Tabela de comandos do ESP8266

Comandos aceitos pelo ESP8266:

BÁSICO			
Nome	**Função**	**Exemplo**	**Observação**
AT	Testa a Comunicação	AT	Retorna OK como resposta
AT+RST	Reseta o módulo	AT+RST	-
AT+GMR	Verificação da versão do módulo	AT+GMR	Retorna a versão do módulo
AT+GSLP	Coloca o módulo em "Sleep Mode"	AT+GSLP=1500	Valor dado em milisegundos
ATE	Habilitar ou Desabilitar echo	ATE	0 = Habilita Echo 1 = Desabilita Echo
CAMADA WIFI			
Nome	**Função**	**Exemplo**	**Observação**
AT+CWMODE	Seleção do modo de trabalho do módulo (estação e/ou ponto de acesso)	AT+CWMODE=modo	modo: 1 = STA Mode (Modo Estação) ;2 = AP Mode (Modo Ponto de Acesso) ;3 = Dual Mode (Ambos os modos)
AT+CWJAP	Conectar em uma rede wifi	AT+CWJAP="nome da rede","senha"	-
AT+CWLAP	Lista de redes disponíveis	AT+CWLAP	Retorna as rede disponíveis para conexões e as suas proteções
AT+CWQAP	Desconectar da rede wifi atual	AT+CWQAP	-
AT+CWSAP	Configuração de um ponto de acesso	AT+CWSAP="nome da rede","senha",canal,tipo de proteção	tipo de proteção: 0 = Rede Aberta; 1 = WPA_PSK ;2 = WPA2_PSK ;3 = WPA_WPA2_PSK
AT+CWLIF	Lista de estações conectadas na rede criada	AT+CWLIF	Retorna os dispositivos conectados na rede criada
AT+CWDHCP	Habilitar ou Desabilitar DHCP	AT+CWDHCP=modo,habilitar	modo: 0 = Definir módulo como ponto de acesso; 1 = Definir módulo como estação; 2 = Definir módulo como ambos habilitar: 0 = Habilita DHCP ;1 = Desabilita DHCP
AT+CIPSTAMAC	Definir endereço MAC do módulo estação	AT+CIPSTAMAC = "mac"	mac = Número de MAC desejado
AT+CIPAPMAC	Definir endereço MAC do módulo ponto de acesso	AT+CIPAPMAC = "mac"	mac = Número de MAC desejado
AT+CIPSTA	Definir endereço de IP da estação	AT+CIPSTA = "ip"	ip = Número de IP desejado
AT+CIPAP	Definir endereço de IP do ponto de acesso	AT+CIPAP = "ip"	ip = Número de IP desejado
CAMADA TCPIP			
Nome	**Função**	**Exemplo**	**Observação**
AT+CIPSTART	Inicia uma conexão	AT+CIPSTART=id,type,addr,port	id: ID da conexão (0-4) tipo: "TCP" ou "UDP" endereço: "" porta: "333"
AT+CIPSEND	Envio de dados	AT+CIPSEND=id,length	id: ID da conexão tamanho: quantidade de caracteres a ser enviado
AT+CIPCLOSE	Fechamento da conexão criada	AT+CIPCLOSE=id	id da conexão criada
AT+CIFSR	Obter IP local	AT+CIFSR	Retorna IP local do módulo
AT+CIPMUX	Habilitar múltiplas conexões	AT+CIPMUX=modo	modo: 0 = Conexão individual 1 = Múltiplas conexões (Máximo 4)
AT+CIPSERVER	Configuração como Servidor	AT+CIPSERVER=modo,porta	modo: 0 = Deletar servidor (precisa ser seguida de um restart do módulo) ;1 = Criar servidor porta: opcional, a padrão é 333
AT+CIPMODE	Configurar como modo de transferência	AT+CIPMODE=modo	modo: 0 = Normal 1 = Modo unvarnished
AT+CIPSTO	Configurar "Time Out" do servidor	AT+CIPSTO=tempo	Faixa aceita de tempo é de 0 a 7200 segundos
AT+CIUPDATE	Atualizar módulo através da rede	AT+CIUPDATE	É aconselhável a fazê-lo caso tenha conhecimento do módulo.
+IPD	Dados recebidos através da rede	+IPD	Todas as mensagens recebidas pelo módulo são iniciados com essa sigla

Figura 33 – Tabela de comandos disponíveis no ESP8266

Fonte: Autor [1]

[1] Tabela baseada no site: https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/

APÊNDICE B – Código da Central (Arduino)

```
//————-Código Utilizado no Componente "Central"(Arduino)
#include <SoftwareSerial.h> //Inclusão da biblioteca de uma porta virtual
SoftwareSerial esp8266(2, 3); //Comunicação serial do ESP8266 com o Arduino nas portas 2 e 3.

void setup()
{
Serial.begin(9600); //Porta serial do Arduino no Baudrate 9600
esp8266.begin(9600); //Porta serial virtual do ESP8266 no Baudrate 9600
sendData("AT+RST\r\n"); //Reset do módulo
delay(3000) //Delay de 3 segundos
sendData("AT+CWQAP\r\n"); //Disconecta de qualquer rede que ele esteja conectado
delay(3000); //Delay de 3 segundos
sendData("AT+CWMODE=3\r\n"); //Modo de funcionamento como estação e ponto de acesso
delay(3000); //Delay de 3 segundos
sendData("AT+CIPMUX=1\r\n"); //Habilitação do modo múltiplas conexões
delay(3000); //Delay de 3 segundos
sendData("AT+CIPSERVER=1,333\r\n"); //Criando um servidor na porta 333
delay(3000); //Delay de 3 segundos
sendData("AT+CWSAP=ËSP,Ï234567890,5,3\r\n"); //Criando a rede com nome "ESP"e senha "1234567890"
delay(3000); //Delay de 3 segundos
}

void loop() {
if (esp8266.available()) //Função ativada quando a porta do ESP8266 estiver disponível
{
Serial.write(esp8266.read()); //Escreve no monitor serial o que é lido da porta virtual do ESP8266
}
if (Serial.available()) //Função ativada quando o monitor serial estiver disponível
{
esp8266.write(Serial.read()); //Escreve na porta virtual do ESP8266 o que foi lido do monitor serial
```

```
}
}

String sendData(String command) //Função de envio de comandos para o ESP8266
{
String response = ; //Deixa a resposta que será lida em branco
esp8266.print(command); //Escrita do comando desejado
long int time = millis(); //Definição da variável "time"como long int, seu valor é a função millis()

while (esp8266.available()) //Função de enquanto a porta virtual do ESP8266 estiver disponível ele executa a sua função
{
char c = esp8266.read(); //Definição da variável "c"como char e ele lê cada caractere recebido pelo ESP8266
response += c; //A variável "response"que inicialmente está vazia recebe caractere por caractere, concatenando cada palavra da resposta
}
Serial.print(response); //Impressão da resposta completa
return response; //retorna a resposta da variável "response"da função
}
```

APÊNDICE C – Código da Central (Visual Studio)

```
//————-Código Utilizado no Componente "Central"(Visual Studio)
Imports System
Imports System.Threading
Imports System.IO.Ports
Imports System.ComponentModel


Public Class Form1
Dim myPort As Array
Delegate Sub SetTextCallback(ByVal [text] As String) 'Added to prevent threading errors during receiveing of data
Dim linha(6) As Int16
Dim linha1(3) As Char
Dim coluna1(6) As Char
Dim matriz(5, 6) As Char

Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
'TODO: esta linha de código carrega dados na tabela 'Database2DataSet3.Clientes'. Você pode movê-la ou removê-la conforme necessário.
Me.ClientesTableAdapter2.Fill(Me.Database2DataSet3.Clientes)
'TODO: esta linha de código carrega dados na tabela 'Database2DataSet2.Clientes'. Você pode movê-la ou removê-la conforme necessário.
Me.ClientesTableAdapter1.Fill(Me.Database2DataSet2.Clientes)
'TODO: esta linha de código carrega dados na tabela 'Database2DataSet1.Clientes'. Você pode movê-la ou removê-la conforme necessário.
Me.ClientesTableAdapter.Fill(Me.Database2DataSet1.Clientes)

myPort = IO.Ports.SerialPort.GetPortNames()
ComboBox1.Items.AddRange(myPort)

Button2.Enabled = False
```

```
End Sub
Private Sub ComboBox1_Click(sender As System.Object, e As System.EventArgs) Handles
ComboBox1.Click
End Sub
Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles But-
ton1.Click
SerialPort1.PortName = ComboBox1.Text
SerialPort1.BaudRate = ComboBox2.Text
SerialPort1.Open()
Button1.Enabled = False
Button2.Enabled = True
Button4.Enabled = True

End Sub
Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles But-
ton2.Click
RichTextBox2.Clear()
SerialPort1.Write(RichTextBox1.Text & vbCrLf) 'concatenate with \n
End Sub

Private Sub Button4_Click(sender As System.Object, e As System.EventArgs) Handles But-
ton4.Click
SerialPort1.Close()
Button1.Enabled = True
Button2.Enabled = False
Button4.Enabled = False
End Sub

Private Sub SerialPort1_DataReceived(sender As System.Object, e As System.IO.Ports.SerialDataReceivedEv
Handles SerialPort1.DataReceived
ReceivedText(SerialPort1.ReadExisting())
End Sub

Private Sub ReceivedText(ByVal [text] As String) 'input from ReadExisting
Dim separador() As Char
separador = ":"
If Me.RichTextBox2.InvokeRequired Then
Dim x As New SetTextCallback(AddressOf ReceivedText)
Me.Invoke(x, New Object() {(text)})
```

```
Else
Dim palavra As String
Dim palavra1() As String
Me.RichTextBox2.Text = [text] 'append text
System.Threading.Thread.Sleep(500)
palavra = Me.RichTextBox2.Text
palavra1 = Split(palavra, ":")
Dim LastNonEmpty As Integer = -1
For i As Integer = 0 To palavra1.Length - 1
If palavra1(i) <>  Then
LastNonEmpty += 1
palavra1(LastNonEmpty) = palavra1(i)
RichTextBox4.Text = palavra1(i)
End If
Next
ReDim Preserve palavra1(LastNonEmpty)
End If
End Sub
Private Sub Button12_Click(sender As Object, e As EventArgs) Handles Button12.Click
Vaga11.Text = "OCUPADO"
Vaga12.Text = "NÃO VALIDADO"
Vaga12.BackColor = System.Drawing.Color.Red
Vaga12.ForeColor = System.Drawing.Color.White
End Sub

Private Sub Button14_Click(sender As Object, e As EventArgs) Handles Button14.Click
Vaga11.BackColor = System.Drawing.Color.WhiteSmoke
Vaga11.Text = "VAGO"
Vaga12.Text = -"
Vaga12.ForeColor = System.Drawing.Color.Black
Vaga12.BackColor = System.Drawing.Color.WhiteSmoke
End Sub

Private Sub Button13_Click(sender As Object, e As EventArgs) Handles Button13.Click
Vaga12.Text = "VALIDADO"
Vaga12.BackColor = System.Drawing.Color.DeepSkyBlue
Vaga12.ForeColor = System.Drawing.Color.Black
End Sub
```

```
Private Sub ClientesBindingNavigatorSaveItem_Click(sender As Object, e As EventArgs)
Me.Validate()
Me.ClientesBindingSource.EndEdit()
Me.TableAdapterManager.UpdateAll(Me.Database2DataSet1)
End Sub

Private Sub Button6_Click(sender As Object, e As EventArgs) Handles Button6.Click
nvaga = 1
numvaga = "Vaga A1"
Form2.Show()
End Sub

Private Sub Button7_Click(sender As Object, e As EventArgs) Handles Button7.Click
nvaga = 2
numvaga = "Vaga A2"
Form2.Show()
End Sub

Private Sub Button8_Click(sender As Object, e As EventArgs) Handles Button8.Click
nvaga = 3
numvaga = "Vaga A3"
Form2.Show()
End Sub

Private Sub Button9_Click(sender As Object, e As EventArgs) Handles Button9.Click
nvaga = 4
numvaga = "Vaga B1"
Form2.Show()
End Sub

Private Sub Button10_Click(sender As Object, e As EventArgs) Handles Button10.Click
nvaga = 5
numvaga = "Vaga B2"
Form2.Show()
End Sub

Private Sub Button11_Click(sender As Object, e As EventArgs) Handles Button11.Click
nvaga = 6
numvaga = "Vaga B3"
```

```
Form2.Show()
End Sub

Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

End Sub

Private Sub Button15_Click(sender As Object, e As EventArgs) Handles Button15.Click

If (RichTextBox4.TextLength = 5) Then
Try
Me.ClientesTableAdapter2.Tag(Me.Database2DataSet3.Clientes, RichTextBox4.Text.Substring(2,
3))
Catch ex As System.Exception
System.Windows.Forms.MessageBox.Show(ex.Message)
End Try

If (RichTextBox4.Text.Substring(0, 2) = 11) Then
aux = 1
If (RichTextBox4.Text.Substring(2, 3) = TagTextBox.Text) Then
Vaga12.Text = "VALIDADO"
Vaga12.BackColor = System.Drawing.Color.DeepSkyBlue
Vaga12.ForeColor = System.Drawing.Color.Black
nome = NomeTextBox.Text
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
Vaga11.Text = "OCUPADO"
ElseIf (RichTextBox4.Text.Substring(2, 3) = 100) Then
Vaga11.Text = "OCUPADO"
Vaga12.Text = "NÃO VALIDADO"
Vaga12.BackColor = System.Drawing.Color.Red
Vaga12.ForeColor = System.Drawing.Color.White
nome = "USUÁRIO NÃO CADASTRADO"
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
End If

ElseIf (RichTextBox4.Text.Substring(0, 2) = 10) Then
Vaga11.BackColor = System.Drawing.Color.WhiteSmoke
```

```
Vaga11.Text = "VAGO"
Vaga12.Text = -"
Vaga12.ForeColor = System.Drawing.Color.Black
Vaga12.BackColor = System.Drawing.Color.WhiteSmoke
aux = 0

ElseIf (RichTextBox4.Text.Substring(0, 2) = 21) Then
aux = 1
If (RichTextBox4.Text.Substring(2, 3) = TagTextBox.Text) Then
Vaga22.Text = "VALIDADO"
Vaga22.BackColor = System.Drawing.Color.DeepSkyBlue
Vaga22.ForeColor = System.Drawing.Color.Black
nome = NomeTextBox.Text
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
Vaga21.Text = "OCUPADO"
ElseIf (RichTextBox4.Text.Substring(2, 3) = 100) Then
Vaga21.Text = "OCUPADO"
Vaga22.Text = "NÃO VALIDADO"
Vaga22.BackColor = System.Drawing.Color.Red
Vaga22.ForeColor = System.Drawing.Color.White
nome = "USUÁRIO NÃO CADASTRADO"
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
End If

ElseIf (RichTextBox4.Text.Substring(0, 2) = 20) Then
Vaga21.BackColor = System.Drawing.Color.WhiteSmoke
Vaga21.Text = "VAGO"
Vaga22.Text = -"
Vaga22.ForeColor = System.Drawing.Color.Black
Vaga22.BackColor = System.Drawing.Color.WhiteSmoke
aux = 0

ElseIf (RichTextBox4.Text.Substring(0, 2) = 31) Then
aux = 1
If (RichTextBox4.Text.Substring(2, 3) = TagTextBox.Text) Then
Vaga32.Text = "VALIDADO"
Vaga32.BackColor = System.Drawing.Color.DeepSkyBlue
```

```
Vaga32.ForeColor = System.Drawing.Color.Black
nome = NomeTextBox.Text
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
Vaga31.Text = "OCUPADO"
ElseIf (RichTextBox4.Text.Substring(2, 4) = 100) Then
Vaga31.Text = "OCUPADO"
Vaga32.Text = "NÃO VALIDADO"
Vaga32.BackColor = System.Drawing.Color.Red
Vaga32.ForeColor = System.Drawing.Color.White
nome = "USUÁRIO NÃO CADASTRADO"
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
End If

ElseIf (RichTextBox4.Text.Substring(0, 2) = 30) Then
Vaga31.BackColor = System.Drawing.Color.WhiteSmoke
Vaga31.Text = "VAGO"
Vaga32.Text = -"
Vaga32.ForeColor = System.Drawing.Color.Black
Vaga32.BackColor = System.Drawing.Color.WhiteSmoke
aux = 0

ElseIf (RichTextBox4.Text.Substring(0, 2) = 41) Then
aux = 1
If (RichTextBox4.Text.Substring(2, 3) = TagTextBox.Text) Then
Vaga42.Text = "VALIDADO"
Vaga42.BackColor = System.Drawing.Color.DeepSkyBlue
Vaga42.ForeColor = System.Drawing.Color.Black
nome = NomeTextBox.Text
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
Vaga41.Text = "OCUPADO"
ElseIf (RichTextBox4.Text.Substring(2, 4) = 100) Then
Vaga41.Text = "OCUPADO"
Vaga42.Text = "NÃO VALIDADO"
Vaga42.BackColor = System.Drawing.Color.Red
Vaga42.ForeColor = System.Drawing.Color.White
nome = "USUÁRIO NÃO CADASTRADO"
```

```
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
End If

ElseIf (RichTextBox4.Text.Substring(0, 2) = 40) Then
Vaga41.BackColor = System.Drawing.Color.WhiteSmoke
Vaga41.Text = "VAGO"
Vaga42.Text = -"
Vaga42.ForeColor = System.Drawing.Color.Black
Vaga42.BackColor = System.Drawing.Color.WhiteSmoke
aux = 0

ElseIf (RichTextBox4.Text.Substring(0, 2) = 51) Then
aux = 1
If (RichTextBox4.Text.Substring(2, 3) = TagTextBox.Text) Then
Vaga52.Text = "VALIDADO"
Vaga52.BackColor = System.Drawing.Color.DeepSkyBlue
Vaga52.ForeColor = System.Drawing.Color.Black
nome = NomeTextBox.Text
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
Vaga51.Text = "OCUPADO"
ElseIf (RichTextBox4.Text.Substring(2, 3) = 100) Then
Vaga51.Text = "OCUPADO"
Vaga52.Text = "NÃO VALIDADO"
Vaga52.BackColor = System.Drawing.Color.Red
Vaga52.ForeColor = System.Drawing.Color.White
nome = "USUÁRIO NÃO CADASTRADO"
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
End If

ElseIf (RichTextBox4.Text.Substring(0, 2) = 50) Then
Vaga51.BackColor = System.Drawing.Color.WhiteSmoke
Vaga51.Text = "VAGO"
Vaga52.Text = -"
Vaga52.ForeColor = System.Drawing.Color.Black
Vaga52.BackColor = System.Drawing.Color.WhiteSmoke
aux = 0
```

```
ElseIf (RichTextBox4.Text.Substring(0, 2) = 61) Then
aux = 1
If (RichTextBox4.Text.Substring(2, 3) = TagTextBox.Text) Then
Vaga62.Text = "VALIDADO"
Vaga62.BackColor = System.Drawing.Color.DeepSkyBlue
Vaga62.ForeColor = System.Drawing.Color.Black
nome = NomeTextBox.Text
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
Vaga61.Text = "OCUPADO"
ElseIf (RichTextBox4.Text.Substring(2, 3) = 100) Then
Vaga61.Text = "OCUPADO"
Vaga62.Text = "NÃO VALIDADO"
Vaga62.BackColor = System.Drawing.Color.Red
Vaga62.ForeColor = System.Drawing.Color.White
nome = "USUÁRIO NÃO CADASTRADO"
matriz1(RichTextBox4.Text.Substring(0, 1), 1) = nome
matriz1(RichTextBox4.Text.Substring(0, 1), 2) = DateTime.Now
End If

ElseIf (RichTextBox4.Text.Substring(0, 2) = 60) Then
Vaga61.BackColor = System.Drawing.Color.WhiteSmoke
Vaga61.Text = "VAGO"
Vaga62.Text = -"
Vaga62.ForeColor = System.Drawing.Color.Black
Vaga62.BackColor = System.Drawing.Color.WhiteSmoke
aux = 0

End If
End If

End Sub

Private Sub TagToolStripButton_Click(sender As Object, e As EventArgs) Handles TagTo-
olStripButton.Click
Try
Me.ClientesTableAdapter2.Tag(Me.Database2DataSet3.Clientes, TagToolStripTextBox.Text)
Catch ex As System.Exception
```

```
System.Windows.Forms.MessageBox.Show(ex.Message)
End Try

End Sub
End Class

Module Module1
Friend nome As String
Friend dataehora As DateTime
Friend matriz1(6, 4) As Object
Friend nvaga As Int16
Friend aux As Int16
Friend numvaga As String
End Module

Public Class Form2

Private Sub ClientesBindingNavigatorSaveItem_Click(sender As Object, e As EventArgs)
Me.Validate()
Me.ClientesBindingSource.EndEdit()
Me.TableAdapterManager.UpdateAll(Me.Database2DataSet1)
End Sub

Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
Label1.Text = "Vaga A1"
If aux = 1 Then
matriz1(1, 1) = "Douglas Kendi Miashiro"
matriz1(1, 2) = DateTime.Now
matriz1(1, 3) = "47.050.219-8"
matriz1(1, 4) = "(11) 96591-0579"
Label4.Text = matriz1(nvaga, 2)
Label3.Text = matriz1(nvaga, 1)
Label9.Text = matriz1(nvaga, 3)
Label8.Text = matriz1(nvaga, 4)
Else
matriz1(1, 1) = "Douglas Kendi Miashiro"
matriz1(1, 2) = DateTime.Now
matriz1(1, 3) = "47.050.219-8"
matriz1(1, 4) = "(11) 96591-0579"
```

```
Label4.Text = matriz1(nvaga, 2)
Label3.Text = matriz1(nvaga, 1)
Label9.Text = matriz1(nvaga, 3)
Label8.Text = matriz1(nvaga, 4)
End If
End Sub

End Class
```

APÊNDICE D – Código do Módulo Fiscalizador (Arduino)

```
//————-Código Utilizado no Componente "Módulo Fiscalizador"(Arduino)
#include <Ultrasonic.h> //Inclusão da biblioteca do Sensor Ultrassônico
#include <AddicoreRFID.h> //Inclusão da biblioteca do leitor RFID
#include <SPI.h> //Inclusão da biblioteca SPI (Serial Peripheral Interface) para comunicação com os periféricos
#include <SoftwareSerial.h> //Inclusão da biblioteca de uma porta virtual

SoftwareSerial esp8266(8, 7); //Comunicação serial do ESP8266 com o Arduino nas portas 7 e 8.

#define uchar unsigned char //Definição de variável uchar como unsigned char
#define uint unsigned int //Defnição de variável uint como usigned int
#define DEBUG true //Defnição da variável DEBUG como TRUE
#define TRIGGER 6 //Definição de TRIGGER como 3
#define ECHO 5 //Definição de ECHO como 4
#define LEDVERMELHO 2
#define LEDAMARELO 3
#define LEDVERDE 4

uchar serNumA[5]; //Array serNumA como unsigned char
uchar fifobytes; //Definição da variável fifobytes como unsigned char
uchar fifoValue; //Definição da variável fifoValue como unsigned char

AddicoreRFID myRFID; //criação do objeto AddicoreRFID para controlar o módulo RFID
const int chipSelectPin = 10; //Porta do módulo RFID como 10
const int NRSTPD = 9; //Porta do módulo RFID como 9

int PRESENCACARRO = 0, TEMPO, AUX = 0, AUX1 = 0, AUXVAL = 5, AUXVAL1 = 0, MINHATAG;
String response = ;
bool VALIDACAO = LOW;
int ESTADOLEDAMARELO = LOW;
int ESTADOLEDVERMELHO = LOW;
```

```
unsigned long millisanterior = 0;
unsigned long millisatual = 0;
long delay5s = 5000;
float DISTANCIACARRO, DISTANCIACM = 20, DISTANCIACMANTERIOR; //Definições de variáveis

Ultrasonic ultrasonic(TRIGGER, ECHO); //Definição das portas do sensor ultrassônico para TRIGGER e ECHO

void setup()
{
Serial.begin(9600); //Porta serial do Arduino no Baudrate 9600
esp8266.begin(9600); //Porta serial virtual do ESP8266 no Baudrate 9600
//delay(3000); //Delay de 3 segundos
//sendData("AT+RST\r\n"); //Reset do módulo
//delay(3000); //Delay de 3 segundos
//sendData("AT+CWMODE=1\r\n"); //Modo de funcionamento como estação
//delay(3000);//Delay de 3 segundos
//sendData("AT+CIPMUX=1\r\n"); //Habilitação do modo múltiplas conexões
//delay(3000);//Delay de 3 segundos
//sendData("AT+CIPSERVER=1,333\r\n"); //Criando um servidor na porta 333
//delay(6000);//Delay de 3 segundos
//sendData("AT+CWJAP=\"ESP\",\"1234567890\"\r\n"); //Conectando a rede com nome "ESP"e senha "1234567890"
//delay(3000);//Delay de 6 segundos

SPI.begin();
pinMode(LEDAMARELO, OUTPUT); //Definição de LEDAMARELO como OUTPUT
pinMode(LEDVERMELHO, OUTPUT); //Definição de LEDVERMELHO como OUTPUT
pinMode(LEDVERDE, OUTPUT); //Definição de LEDVERDE como OUTPUT
pinMode(chipSelectPin, OUTPUT); //Definição do chipSelectPin como OUTPUT
digitalWrite(chipSelectPin, LOW); //Ativação do pino chipSelectPing
pinMode(NRSTPD, OUTPUT); //Definição de NRSTPD como OUTPUT
digitalWrite(NRSTPD, HIGH); //Ativação do pino NRSTPD
myRFID.AddicoreRFID_Init(); //Iniciação do leitor RFID
}

void loop()
{
```

```
DISTANCIACARRO = CARROPRESENTE(); //Uso da função CARROPRESENTE para definir a distância de um carro
VAGALIVRE(); //Envio de dados do Módulo Fiscalizador para a Central indicando que a vaga está vazia
while (DISTANCIACARRO > 10) //Enquanto a distância do carro for menor que 10 cm (Vaga livre)
{
AUX1 = 0;
AUX = 0;
AUXVAL = 0;
VALIDACAO = LOW;
digitalWrite(LEDAMARELO, LOW); //Led amarelo desligado
digitalWrite(LEDVERMELHO, LOW); //Led vermelho desligado
digitalWrite(LEDVERDE, LOW); //Led verde desligado
DISTANCIACARRO = CARROPRESENTE(); //Verificação da distância novamente
}
while (DISTANCIACARRO < 10) //Enquanto a distância do carro for menor que 10 cm (Vaga ocupada)
{
VAGAOCUPADA();
digitalWrite(LEDAMARELO, HIGH); //Led Amarelo ligado
DISTANCIACARRO = CARROPRESENTE(); //Verificação da distância
VALIDACAO = RFID(); //Chama função de validação de dados (loop infinito de validação até ser validado)
if (AUX1 == 0) //Variável auxiliar para contagem de tempo de validação excedido
{
millisanterior = millis();
millisatual = millis();
AUX1 = 1;
}
if (millisatual - millisanterior > delay5s) //Caso passar o tempo de validação, se muda a auxiliar para entrar em outro laço da função
{
AUX = 1;
}
else
{
millisatual = millis(); //Caso o tempo não tiver excedido, ele atualiza o tempo para acumular a contagem
```

```
}
while (DISTANCIACARRO < 10 & VALIDACAO == LOW & AUX == 1) //Caso o carro ainda estiver presente, não houver validação e a auxiliar tiver o valor de 1, se considera que o tempo de validação foi excedido
{
digitalWrite(LEDVERMELHO, HIGH); //Led Vermelho ligado
digitalWrite(LEDAMARELO, LOW); //Led Amarelo desligado
digitalWrite(LEDVERDE, LOW); //Led Verde desligado
DISTANCIACARRO = CARROPRESENTE(); //Verificação da distância
VALIDACAO = RFID(); //Chama função de validação de dados (loop infinito de validação até ser validado)
if (VALIDACAO == HIGH) //Caso validação for verificada
{
digitalWrite(LEDVERMELHO, LOW); //Led Vermelho desligado
digitalWrite(LEDAMARELO, LOW); //Led Amarelo desligado
digitalWrite(LEDVERDE, HIGH); //Led Verde ligado
}
}

while (DISTANCIACARRO < 10 & VALIDACAO == HIGH) //Loop infinito até o carro sair da vaga (distancia maior que 10cm)
{
digitalWrite(LEDVERMELHO, LOW); //Led Vermelho desligado
digitalWrite(LEDAMARELO, LOW); //Led Amarelo desligado
digitalWrite(LEDVERDE, HIGH); //Led Verde ligado
DISTANCIACARRO = CARROPRESENTE(); //Verificação da distância
}
}
}


float CARROPRESENTE()
{
long microsec = ultrasonic.timing();
DISTANCIACMANTERIOR = DISTANCIACM;
DISTANCIACM = ultrasonic.convert(microsec, Ultrasonic::CM);
if (DISTANCIACM - DISTANCIACMANTERIOR >= 10)
{
return DISTANCIACM;
```

```
}
}

bool RFID()
{
{
uchar i, tmp, checksum1;
uchar status;
uchar str[MAX_LEN];
uchar RC_size;
uchar blockAddr; //Selection operation block address 0 to 63
String mynum = ;

status = myRFID.AddicoreRFID_Request(PICC_REQIDL, str);

//Anti-collision, return tag serial number 4 bytes
status = myRFID.AddicoreRFID_Anticoll(str);
if (status == MI_OK)
{
// Serial.println("RFID tag detected");

if (AUXVAL == 1) {
AUXVAL = 2;
MINHATAG = 11;
if (str[0] == 105) {
VALIDACAO = HIGH;
sendData("AT+CIPCLOSE=0\r\n"); // rst
delay(3000);
sendData("AT+CIPMUX=1\r\n"); // rst
delay(3000);
sendData("AT+CIPSTART=0,\"TCP\",\"192.168.4.1\",333\r\n"); // rst
delay(3000);
sendData("AT+CIPSEND=0,5\r\n"); // rst
delay(3000);
sendData("11105");
}
MINHATAG = 0;
MINHATAG = 11000 + str[0];
}
```

```
// while (esp8266.available())
// {
// char c = esp8266.read(); // read the next character.
// response += c;
// } // Serial.println();
// if (response == "+IPD(0,2):OK")
// {
// VALIDACAO = HIGH;
// }
}
// sendData("AT+CIPCLOSE=0\r\n"); // rst
myRFID.AddicoreRFID_Halt(); //Command tag into hibernation
}
return VALIDACAO;
}

void VAGAOCUPADA()
{
if (AUXVAL == 0) {
AUXVAL = 1;
sendData("AT+CIPCLOSE=0\r\n"); // rst
delay(3000);
sendData("AT+CIPMUX=1\r\n"); // rst
delay(3000);
sendData("AT+CIPSTART=0,\"TCP\",\"192.168.4.1\",333\r\n"); // rst
delay(3000);
sendData("AT+CIPSEND=0,5\r\n"); // rst
delay(3000);
sendData("11000");
}
}

void VAGALIVRE()
{
}

String sendData(String command)
{
```

```
if (AUXVAL == 0) {
AUXVAL = 1;
sendData("AT+CIPCLOSE=0\r\n"); // rst
delay(3000);
sendData("AT+CIPMUX=1\r\n"); // rst
delay(3000);
sendData("AT+CIPSTART=0,\"TCP\",\"192.168.4.1\",333\r\n"); // rst
delay(3000);
sendData("AT+CIPSEND=0,5\r\n"); // rst
delay(3000);
sendData("10000");
}
```